Publication No.: 10-2001-0109271
Applicant: INFRAWORKS CORPORATION

ABSTRACT

A system and method of protects security of data. The data is packaged together with one or more permission that designate what actions are allowed with respect to the data. The package can be opened when there is approval for doing so and the allowed permissions are maintained. The data is stored within a vault and there are a number of available security procedures that prevent the unauthorized access of the data.

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(51) 。Int. Cl. [7]
G06F 11/00

(11) 공개번호 특2001-0109271
(43) 공개일자 2001년12월08일

(21) 출원번호 10-2001-7006901
(22) 출원일자 2001년06월01일
번역문 제출일자 2001년06월01일
(86) 국제출원번호 PCT/US2000/26882
(86) 국제출원출원일자 2000년09월29일
(87) 국제공개번호 WO 2001/25870
(87) 국제공개일자 2001년04월12일

(81) 지정국 국내특허 : 탄자니아, 모잠비크,
AP ARIPO특허: 가나, 감비아, 케냐, 레소토, 말라위, 수단, 시에라리온, 스와질랜드, 우간다, 짐바브웨,
EA 유라시아특허: 아르메니아, 아제르바이잔, 벨라루스, 키르기즈, 카자흐스탄, 몰도바, 러시아, 타지키스탄, 투르크메니스탄,
EP 유럽특허: 오스트리아, 벨기에, 스위스, 독일, 덴마크, 스페인, 프랑스, 영국, 그리스, 아일랜드, 이탈리아, 룩셈부르크, 모나코, 네덜란드, 포르투칼, 스웨덴, 핀랜드, 사이프러스,
OA OAPI특허: 부르키나파소, 베넹, 중앙아프리카, 콩고, 코트디브와르, 카메룬, 가봉, 기네, 말리, 모리타니, 니제르, 세네갈, 차드, 토고, 기네비쏘,

(30) 우선권주장 60/157,472 1999년10월01일 미국(US)
60/206,947 2000년05월25일 미국(US)

(71) 출원인 인프라워크스 코포레이션
추후제출
미국 텍사스 78701 오스틴 라바카스트리트 504 스위트 1100

(72) 발명자 프리드만조오지
미국텍사스78731오스틴몬타나노트7109
스타렉로버트필립
미국텍사스78727오스틴델로블스3609
머독카를로스
미국텍사스78751오스틴에비뉴에프4517

(74) 대리인 임훈빈

심사청구 : 없음

(54) 데이터보안 제공을 위한 시스템과 방법

요약

본 발명은 파일의 뒤이은 전송을 보호하거나 비권한자가 파일을 열어보는 것을 막는 데이터보안시스템과 방법을 제공한다.

대표도
도 1

색인어
보안, 데이터보안

명세서

기술분야

본 발명은 컴퓨터에 저장되는 데이터의 보호에 관한 것이다. 특히, 외부의 소스로부터 전송받은 데이터의 보호에 관한 것이다.

배경기술

데이터는 씨디롬, 플로피디스크 또는 다른 입출력장치로부터의 다운로드, 인터넷과 같은 전자전송과 인터넷전송을 포함하는 여러 소스로부터 컴퓨터에 로드될 수 있다. 일단 이 파일들은 다운로드되면, 상기 파일의 전송자나 제작자가 그 정보를 제어할 수 있는 방법이 없다. 기밀문서와 적당한 내용이 포함된 자료가 이러한 방법으로 전송되고 있기 때문에 이러한 문제는 매우 중요시 되어왔다.

본 발명은 전술된 관점에서 상기와 같은 문제점을 극복하고자 개발되었다.

발명의 상세한 설명

상세하게 도면을 참조하면, 여러 도면에서 같은 도면부호는 같은 요소로 일치하며, 도 1은 데이터보안시스템(10)을 구성하는 데이터보안을 위한 시스템과 방법의 일 실시예를 나타낸다. 본 발명의 실시예로서, 도시된 데이터보안시스템(10)은 바람직하게 패키저(12)와, 클라이언트 또는 리시버(14)로 구성된다. 본 명세서상에서 상세히 설명될 것으로서, 본 발명의 실시예의 패키저(12)는 안전한 데이터문서를 만들고 상기 리시버(14)가 상기 안전한 데이터에 대한 이용을 관리한다. 바람직한 실시예로서, 패키저(12)와 리시버(14)는 데이터보안을 만들고 유지하기위하여 컴퓨터 네트워크상에서 결합하여 작동한다. 도시되진 않았지만, 다른 실시예로서, 바람직하게 패키저(12)와 리시버(14)는 서로 독립적으로 이용될 수 있다. 본 명세서상에서 언급되는 " 컴퓨터" , " 컴퓨터시스템" , 또는 " 시스템" 은 대체로 정보를 수신, 송신 그리고/또는 이용가능하게 하는 장치를 포함하는 것으로 설명될 수 있다. 상기 장치는 제한없이 프로세서, 마이크로프로세서 또는 그와 유사한 디바이스, 개인용 컴퓨터(예를 들어 랩톱, 팜PC), 데스크탑 또는 워크스테이션, 네트워크서버, 메인프레임, 유무선기기가 있다. 상기 유무선기기의 예로서 전화와, 상호작용 텔레비젼, 셀룰러폰, 개인용 디지털 보조장치, 무선호출기, 디지털시계가 있다. 상기 상호작용 텔레비젼으로는 인터넷에 연결되어 사용되는 텔레비젼 또는 텔레비젼과 함께 사용되는 전자기기가 있다. 실예로서, 정보는 전자우편(e-mail) 형태로 전송된다. 게다가 본 발명의 컴퓨터, 컴퓨터시스템, 또는 시스템은 네트워크상에서, 예를 들어 인터넷, 인트라넷, 엑스트라넷상에서, 다른 시스템과 통신으로 작동될 수 있고, 또는 고립된 시스템으로도 작동될 수 있다.

바람직한 실시예로서 패키저(12)는 수신자에게 안전하게 데이터를 전할 수 있도록 실행가능한 것을 생성시킬 수 있도록 작동된다. 리시버(14)는 바람직하게 하나 이상의 디바이스 드라이버와 안전한 데이터의 사용을 관리하는 응용프로그래밍 인터페이스(예를 들어 Win32 프로세서)로 구성된다. 본 명세서에서 상세하게 기술될 것으로서, 리시버(14)부의 실시예로서 커널-모드 모듈로 구성되는 아홉개의 디바이스 드라이버가 있다. 본 명세서에서 언급되는 " 데이터" 또는 " 정보" 는 교체될 수 있고 대체로 정보의 유형, 데이터 또는 다른 내용(예를 들어 텍스트, 그래픽, 오디오, 비디오

또는 적당한 컴퓨터 파일포맷 같이 요구되는 포맷에서의 어떤 결합)을 포함하도록 설명될 수 있다.

보호된 정보의 발송인이 독점되어 있거나 기밀의 데이터의 패키지를 외부의 소스에 보낼려고 할 때, 패키지된 파일은 스크린 프롬프트에 응답하여 선택된다.

데이터 패키지의 작동에 대한 승인 룰 셋이 그 때에 선택된다. 이러한 승인 룰 셋은 여러가지 다른 타입중 어느 것도 될 수 있으며, 예를 들어 특히 복사방지와 같이 실행되는 기능, 허락된 복사의 횟수(예를 들어 2) 또는 소유에 대한 시간 제한(예를 들어 1시간)이 있다. 이는 기초 승인 룰이 되거나 재설정될 수 있다. 패키지는 그 안에 데이터와 룰을 가진 실행가능한 번들을 생성시키므로서 구성된다. 본 실시예에 따른 패키저(12)의 바람직한 실시예는 본 명세서상에서 기술될 것이다.

최종사용자가 패키저(12)로부터 데이터를 받을 때, 최종사용자는 파일을 열고 실행가능한 패키지에서 감지프로그램은 바람직하게 리시버(14)가 작동하고 있는지를 확인하기위하여 검사한다. 그리고 그때 승인이되면 패키지를 처리하는 리시버(14)에 신호를 보낸다. 이러한 실시예에서, 감지프로그램은 실행가능한 패키지를 읽을 수 있는 드라이버를 찾는다. 상기 리시버(14)는 본 실시예에서 적어도 하나의 디바이스 드라이버와 바람직하게는 아홉개의 디바이스 드라이버로 구성되며, 이에 대한 내용을 아래에서 더 상세하게 기술될 것이다. 도시되지 않은 다른 실시예로, 더 크거나 더 적은 수의 디바이스 드라이버와/또는 디바이스 드라이버의 다른 유형도 바람직하게 이용될 수 있다. 이러한 설명의 목적을 위하여, 다음에서 첫머리에 하나의 디바이스 드라이버에 대한 실시예에서 아홉개의 디바이스 드라이버의 기능적인 면을 기술할 것이다. 왜냐하면 디바이스 드라이버의 수는 변화할 수 있기 때문이다. 일반적으로, 이러한 실시예에서 다운로드된 파일의 보안은 안전한 데이터파일이 포함된 공동의 저장소와 보호용 드라이버의 사용을 통한 리시버(14)에 의해 유지된다. 보호되어야 하는 모든 데이터파일은 바람직하게 상기 저장소에 저장된다. 사실상, 상기 저장소는 보호되는 파일의 저장소이다. 만약 상기 드라이버가 발견될 수 없다면, 상기 패키지는 사용자에게 드라이버가 필요하다고 알린다. 그리고/또는 그때문에 자체적으로 파괴되어 자체적으로 덮어쓰거나 또는 몇 가지 다른 요구되는 선택사항을 제공한다. 드라이버를 찾았다면, 그때 드라이버는 상기 저장소에 저장된 데이터로부터 규칙을 세밀히 조사하고, 컴퓨터 이용에 대한 "태그" 파일을 생성하며, 실행가능한 파일과 데이터의 모든 흔적을 제거하기 위하여 상기 저장소의 외부에 있는 실행가능한 파일을 덮어쓴다(갈가리 찢는다). 이 실시예에서, 드라이버는 상기 드라이버가 컴퓨터 프로세스에 대한 위치 홀더로 작용하는 저장소에 저장되는 가상의 테이블을 이용하여 실제의 파일에 나타내는 태그파일을 생성한다.

본 실시예에서, 상기 저장소는 하드드라이브의 공간이고 아래에 기술된 저장소제어블록, 파일제어블록, 저장소할당테이블, 저장소데이터블록과 같은 내부 보안도 마찬가지로 보호되는 파일을 포함한다. 이 실시예에서, 상기 저장소에 포함된 파일들은 상기 파일, 생성된 드라이버 또는 상기 파일과 상기 실제 데이터파일 그 자체에 대한 이름이 생성된 데이터 생성자에 대한 컴퓨터 이름을 포함한다. 저장소할당테이블은 컴퓨터 이름과 상기 파일에 할당되고 상기 저장소에 저장되는 이름을 연결시키기 위한 드라이버로 어드레스할 수 있는 연결 테이블로서 작동된다. 게다가, 이러한 실시예에서 상기 저장소에 있는 상기 데이터는 바람직하게 암호화되고/또는 압축될 수 있으며, 늘려질 수 있으며 더 나은 보안을 제공하기 위하여 복잡화될 수 있다. 또한, 상기 저장소는 바람직하게 외부 보호에 대하여 여러 레벨로 보호되며, 아래에서 상세히 기술된다.

상기 드라이버는 다운로드될 때 컴퓨터 사용자와 상기 저장소 사이에 스스로 위치하는 저장소 특정 드라이버이다. 상기 저장소와 드라이버는 하드드라이브에 로드된다. 상기 드라이버는 컴퓨터에서 낮은 레벨 호출을 감시하고 상기 가상테이블을 이용하여 저장소의 파일을 작동시키며 내용에 상관없이 파일시스템을 보호하기 위하여 컴퓨터 사용자로부터 입력한다. 상기 드라이버는 컴퓨터의 부트에 로드된다. 드라이버는 자신의 위치와 저장소의 위치를 알고, 상기 드라이버는 그 자체적으로 할당하고 컴퓨터 운영체제 사용자에 의해 이용되는 저장소의 새로운 태그 이름을 할당한다. 상기 컴퓨터

의 하드드라이브에 존재할 때 상기 컴퓨터는 이 태그(스푸프(spoofed)태그)이름에 작동되고 실제 파일이름이나 드라이브의 위치는 모르기 때문에, 상기 드라이버와 저장소의 존재, 위치는 시스템 사용자에게 보여질 수 없거나 " 은폐" 된다. 그것들은 볼 수 없기 때문에, 드라이버와 저장소의 존재와 위치는 컴퓨터 사용자에 의해 부트이후에 어드레스될 수 없다.

상기 파일이 파일을 저장하는 저장소의 태그 또는 드라이버 자체와 일치할 때, 상기 드라이버는 바람직하게 막기위한 명령이 사용자로부터 명시된 파일을 접근하는 것을 차단한다.

상기 드라이버는 하드드라이브이고 메모리에 부트된다. 상기 드라이버는 컴퓨터가 은폐된 드라이버를 부트하도록 실행해서 먼저 첫번째 드라이버가 로드되고 상기 컴퓨터가 은폐된 드라이버를 닫게하는 원인이 된다. 먼저 초기화 또는 부트되기 때문에, 상기 드라이버는 항상 작동되고 있고 그러므로 상기 드라이버와 저장소와 상기 저장소 사이의 요소를 항상 은폐시킬 수 있다. 이러한 방법으로, 상기 요소는 항상 보호되고 작동에 대해 준비되어 있다.

드라이버는 첫번째이기 때문에, 드라이버는 항상 은폐되도록 요구되는 것을 은폐할 수 있다. 이것은 시스템에서 해크되는 파일을 위치시키는 것을 어렵게 만드는 방어의 첫번째 레벨에 있다. 은폐된 드라이버가 먼저 걸리지 않으면, 그때 이것은 권한없는 사용자가 상기 드라이버나 상기 저장소 파일에 포함된 기밀의 데이터를 해크하거나 손상시키려는 매우 좋은 암시이다. 컴퓨터가 부트될 때, 그것은 파일시스템에 걸린다. 혹(hook)은 동일함을 증명할 수 있고 드라이버는 상기 혹을 감지하고 그들을 카운트할 수 있다. 상기 드라이버는 그것이 첫번째 혹이 아닌 것을 감지한다면, 그때 상기 드라이버는 바람직하게 존재하는 혹을 걸기위하여 시도한다. 다른 실시예로서, 시스템은 상기 드라이브가 위치된 하드드라이브부를 급속히 덧쓰므로서 자체적으로 그리고 상기 저장소를 손상시키므로서 배열될 수 있다. 이러한 방법으로 상기 데이터는 함부로 변경될 수 없다.

작동하는 동안, 상기 드라이버는 태그 또는 실제의 컴퓨터 운영체제에 대한 위치 홀더를 생성한다. 또한 상기 태그는 태그이름을 실제 이름으로 변환하는 드라이버에서 엔트리(entry)이다. 상기 드라이버는 상기 저장소 내의 파일에서 프로세스가 수행되는지 안되는지를 결정하기위하여 운영체제에 대한 낮은 레벨 호출을 차단한다. 상기 프로세스가 저장소 내의 파일에서 수행된다면, 그때 상기 드라이버는 프로세스가 확인된 파일에서 수행되는지 그렇지 않은지를 결정하기 위하여 저장소할당테이블에서 기능 그리고 실제 파일이름과 승인 테이블을 비교한다. 승인이 거부되면, 그때 상기 드라이버는 파일이 검색될 수 없거나 허용되지 않는 명령이 수행되고 프로세싱을 방해하는 WIN32 계층을 통하여 메세지를 리턴하거나, 덮어쓰기를하므로서 자체적으로 손상시킬 수 있다. 승인이 허락되면, 그대 상기 드라이버는 특정 파일의 저장소 주소에 대해 상기 테이블로 이동한다. 상기 드라이버는 저장소로부터 파일을 빼내기 위하여 주소, 암호화의 압축, 저장소할당테이블의 패딩정보를 이용하며 설명된 바와 같이 작동된다. 상기 저장소제어블록은 파일에 대해 허가된 횟수 또는 사용의 지속기간에 관하여 룰 셋에 따를지를 결정하기 위하여 상기 파일이 어떻게 이용되는지를 감시한다.

해커는 부팅 전에 시스템에 침입할 수 있고 저장소에 위치될 수 있다. 그러므로, 보호 층(layer)은 파일이나 드라이버의 권한이 없는 복사, 이동, 전송 또는 조작하는 것을 막기위해서 제공된다. 따라서, 상호적으로 독립적이고 계층이 있는 보호 메카니즘은 상기 드라이버와 저장소에 포함된다.

첫째 메카니즘은 상기 저장소, 드라이버 그리고/또는 승인 데이터베이스를 권한이 있는 컴퓨터에 고정시키는 것이다. 이러한 사항에 대한 상기 고정점은 바람직하게 시스템이 작동되는 첫번째 시간으로 결정된다. 상기 고정점은 상기 저장소, 드라이버 또는 승인 데이터베이스의 시작 클러스터가 위치되는 어드레스로 정의된다. 여러가지 고정은 바람직하게 이러한 실시예에서 결합되고 암호화/복호화에 대한 키로 사용된다. 상기 키가 이 실시예에서 그 다음에 " 작동" 되지 않는다면, 그때 하나 이상의 파일들이 이동되고 수정되는 것은 암시이다. 그러므로, 앞에서 작동되는 바와 같이, 상기 시스템은 드라이버, 저장소와/또는 승인 데이터베이스가 이동되는지를 보기위하여 검사할 것이다. 그렇다면, 이러한 실시예에서, 드라이버가 저장소와 자체적으로 덮어쓰도록 명령할 것이다. 왜냐하면 이는 변경하거나 이동에 대한 표시이기

때문이다. 컴퓨터 하드드라이브의 저장된 파일이나 드라이버의 시작점은 보통 컴퓨터 제작, 컴퓨터 모델, 그리고 컴퓨터 하드드라이브에 이미 저장된 데이터와 파일의 양의 기능일 것이다. 그러므로, 이는 임의 발생자 인식 코드와 동등한 것이다. 상기 드라이버나 저장소가 부트되고 자신의 고정 위치를 인식하지 못한다면, 그 자체적으로 덮어스기가 될 것이다. 게다가, 대응되는 요소(대응되는 드라이버 또는 저장소)의 어드레스가 인식되지 않는다면, 그것 또는 다른 것이 이동되고 그러므로 변경되어 자체적으로 덮어쓰기될 것을 안다.

추가적으로, 상기 드라이버는 CRC 유형의 검사 합계를 저장소에 보낼 수 있다. 상기 저장소는 파일에서 작동을 허락하기전에 검사 합계를 이용하는 드라이버의 보전과 본질을 확인한다. 다른 것을 인식하지 못한다면, 그때 그것들은 상기 파일의 권리가 없는 사용을 막는 효과로 자체적으로 덮어쓸 것이다.

처리되는 동안, 상기 저장소내의 파일 데이터에서 컴퓨터 프로그램 작동은 바람직하게 다른 소프트웨어가 즉시 작동되도록 하는 데이터파일을 생성한다. 한가지 예로서 프린트하는 것이 있다. 프린트 파일은 상기 저장소 내에 중간 데이터 파일을 생성한다. 그래서 " 백-채널링" 은 상기 저장소의 외부에서는 일어나지 않는다. 결과적으로, 사실상 그들의 이용에 대해 임시파일을 생성하는 다른 소프트웨어 프로그램은 상기 저장소에서 그들의 동작을 수행하기 위해서 상기 드라이버를 통하여 작동한다.

상기 저장소와/또는 드라이버는 어떻게 해서 권한이 없는 컴퓨터에서 재전송되는 것이라면, 본 발명은 데이터의 유출을 막기위해서 포트제어시스템 레벨에서 작동되며 아래에서 상세히 기술된다.

그러한 보호를 위한 한 가지 시도는 상기 저장소에 검색되지 않는 파일에 대응되는 보호된 파일을 전체 문서에 복사하는 것이 될 수 있다. 사실, " 클립보드" 는 마이크로소프트 워드로부터 잘 알려진 기능이다. 상기 드라이버는 낮은 레벨 운영체제 호출을 감시한다. 저장된 파일에 대한 복사명령이 상기 드라이버에 의해 감지되었을 때, 상기 드라이버는 메모리를 가득채운 WIN32 계층을 통하여 메세지를 출력하거나 메모리 없음 또는 비정상적인 동작으로 사용자에게 알린다.

상기된 바와 같이, 본 실시예에서 아홉개의 디바이스 드라이버가 있는 것이 바람직하다. 본 실시예에서 상기 아홉개의 디바이스 드라이버는 WIN32 프로세스(DG32(20))와 상기 드라이버 사이에 주요하게 통신할 수 있는 DGCOMM(22); 파일시스템 드라이버로 작동되는 DGFS(24); 승인 데이터베이스와 강제 승인을 유지할 수 있는 DGPRMDB(26); 상기 저장소에 대해 책임을 지고 모든 암호화 데이터를 포함하는 DGVAULT(28); 입출력포트, 레지스트리, TCP/IP와 분할된 메모리에 접근하는 것을 보호하는 DGGUARD(30); 정기적으로 시스템 무결성을 검사하는 DGWATCH(32); 규칙적인 시간간격을 제공하는 DGTIMER(34); 구성요소들을 이동하거나 해킹하는 것을 방지하는 DGANCHOR(36); 데이터의 암호화/복호화를 제공하는 DGCRYPT(38)로 도 1에서 확인된다.

DG32(20)는 리시버(14)에 대해 명령센터의 역할을 하는 WIN32 프로세스이다.

이 실시예에서, DG32(20)는 실행되는 패키지를 관찰하고 사용자에게 파일이름을 패키지에 저장하라고 재촉하고; 패키지 패이로드(payload)가 상기 저장소에 흡수되는 DGFS(24)에 알리고; DGFS(24)가 애플리케이션이 보호된 파일을 열려고 시도하고 있는 DG32(20)에 알릴 때, 다이알로그로 최종사용자에게 알리고; 애플리케이션이 나갈 때 디바이스 드라이버에 알리고; 승인이 만료된 때, 클립보드 접근을 막고 승인 디바이스 드라이버의 요청에 애플리케이션을 종료한다.

DGCOMM(22)는 WIN32 계층과 디바이스 드라이버 계층 사이에서 모든 통신을 다룬다.

이 실시예에서, DGCOMM(22)는 바람직하게 정보를 보내거나 상기 디바이스 드라이버 계층으로부터 정보를 받기위하여 WIN32로부터 호출되는 디바이스 드라이버 입출력 제어기능의 싱글 셋을 제공하고, 상기 디바이스 드라이버 계층으로부터 몇몇 정보를 회수하기 위하여 디바이스를 호출하는 WIN32 계층에 신호를 보내는 다른 디바이스 드라이버로 사용되는 기능을 제공한다.

이 실시예에서 DGFS(24)는 설치할 수 있는 파일시스템관리자를 통하여 지날 수 있는 파일시스템 호출을 유지하도록 하는 것을 가능하게 한다. 상기 DGFS(24)는 바람직하게 애플리케이션이 보호되는 파일에 접근을 강요하는 주요한 기능을 제공하고, 상기 저장소의 내부와 외부로 데이터를 전달하며, 태그 파일에 대해 파일 크기를 속이고 보호된 시스템 파일을 숨기거나 " 은폐" 시킨다.

이 실시예에서 DGVAULT는 DCCRYPT 파일을 여는 애플리케이션으로 생성되는 보호된 파일과 임시파일을 저장하는데 대해 자체적으로 포함된 파일시스템이다. 바람직하게 상기 저장소의 모든 데이터는 암호화되고 저장소제어블록, 파일제어블록, 저장소할상테이블과 저장소데이터블록을 포함하는 구성요소로 구성된다.

이 실시예에서 DGPRMDB(26)는 상기 저장소의 파일에 관한 정보(예를 들어 데이터파일이 열릴 수 있는 시간, 파일이 열릴 수 있는 횟수, 파일이 더이상 열릴 수 없는 자료, 파일 프린트 승인과 클립보드 승인을 포함하는 파일 승인)를 포함한다. 게다가, 승인 데이터베이스는 바람직하게 상기 저장소에 사용되는 파일이름, 열려진 패키지의 패키지아이디와 시스템아이디를 포함한다.

이 실시예에서 DGCRYPT(38)는 상기 시스템에 대하여 암호화와 복호화 서비스를 제공하는데, 예를 들어 보호된 패키지로부터 보호된 파일데이터를 얻기위한 강한 복호화, DGVAULT(28)에 의해 관리되는 저장소의 내부로 들어오거나 밖으로 나가는 데이터에 대한 약한 암호화/복호화가 있다.

이 실시예에서 DGANCHOR(36)는 상기 시스템에 대하여 보안의 다른 계층을 제공한다. 예를 들어, 이 실시예에서 DGANCHOR(36)은 파일의 물리적 위치와 암호화/복호화에 대한 키로 사용되는 모든 앵커(anchor)의 결합에 기초한 시스템에서 각각의 파일에 대한 유일한 동일자(고정)를 결정한다.

이 실시예에서 DGGUARD(30)는 여러가지 레지스트리 입구에 접근을 막을 수 있으며, 예를 들어 직렬과 병렬 포트 연결, 네트워크 연결, 데이터 드래그앤그롭(drag and drop)과 이메일 연결이 있다.

DGWATCH(32)는 상기 시스템에 대해 감시자로서 기능을 한다. 이 실시예에서, 상기 DGWATCH(32)는 시스템 무결성을 보증하기 위해서 바람직하게는 일정한 시간간격으로 시스템의 디바이스 드라이브의 모든 것을 조사한다.

이 실시예에서 DGTIMER(34)는 상기 시스템에 대해 내부 타이머 서비스를 제공한다. 본 실시예에서, DGTIMER(34)는 구성요소의 모든 것이 여전히 적당하게 작동되고 있는 것을 보증하기 위하여 바람직하게 일정한 시간간격으로 DGWATCH(32)를 호출하고, 타이머와 승인이 업데이트될 수 있도록 바람직하게 일정한 시간간격으로 DGPRMBD(26)을 호출하고, 전체 시스템 무결성검사를 하기 위하여 셋업한 이후 짧은 시간간격(수 초)으로 DGPRMBD(26)을 호출한다. 본 실시예에서 아홉개의 디바이스 드라이버의 특성과 작동에 있어 여러 디바이스 드라이버들 사이의 관계는 아래에서 더 상세하게 기술된다.

본 실시예에 따른 다른 보호 메카니즘은 시스템 레지스트리를 감시하는 것이다. 레지스트리는 구성 데이터에 대한 계층적인 저장소이다. 상기 레지스트리는 컴퓨터상에 저장된 모든 소프트웨어의 트랙과 프로그램들간의 관계를 유지시킨다. 상기 레지스트리는 대다수의 사용자와 하드웨어 구성의 트랙을 유지시킨다. 레퍼런스는 대다수의 사용자 사이에서 변화한다.

상기 레지스트리에서 각각의 데이터는 그와 관계된 키-값을 가진다. 이름과 값은 등록값으로 나타낸다. 키는 폴더와 유사하고, 자체적으로 서브키로 불리는 하나 이상의 폴더와 하나 이상의 이름-값 쌍을 포함할 수 있다. 또한, 상기 키는 이름 또는 핸들로도 나타낸다. 데이터에 접근하고 저장된 값을 수정하기 위해서는 정확한 키가 필요하다.

상기 레지스트리는 데이터베이스이고, 즉 데이터 저장 위치이기 때문에, 누설되는 데이터로 이용될 수 있다. 본 명세서에서 언급된 " 누설데이터" 는 데이터가 보호되는 것이 요구되면서 시스템 외부로 전송되는 데이터를 의미한다. 프로세스는 상기 레지스트리에 정보를, 예를 들어 임시저장소에, 기록한다. 이때 다른 프로세스는 상기 레지스트리로부터 상기 정보에 접근할 수 있으며 상기 데이터는 레지스트리키에 기록한다. 이때 다른 프로세스는 상기 레지스트리키로부터 상기 데이터를 읽을 수 있고 그것을 디스크 또는 다른 저장장치에 기록한다. 즉 이것이 누설데이터인 것이다. 따라서, 애플리케이션에 대해서 데이터보안은 중요하며, 상기 레지스트리로부터 데이터 누출을 제한하는 것이 필요하다.

다음은 상기 레지스트리로부터 데이터 누출을 줄이거나 제거할 수 있는 레지스트리를 감시하기 위한 방법과 시스템의 여러 실시예를 설명한다. 본 발명은 시스템의 보이지 않는 부분에서 데이터 이동을 거절하는 프로세스를 보장한다. 예를 들어, 본 발명은 파일시스템 읽기, 통신포트상에서 데이터이동, 다른 프로세스와의 메모리 공유, 레지스트리에 데이터 기록을 제한하는 것을 포함한다.

레지스트리 감시방법의 전형적인 실시예는 호출프로세스에서 레지스트키에 대한 핸들 요청, 상기 핸들에 대한 레지스트리키 값 요청, 보호되는 프로세스 리스트와 거절 리스트를 검사하므로서 상기 요청을 완료하기 위한 보안해제 획득으로 구성된다. 레지스트리에 추가하는 자격이 차단되기 때문에, 상기 레지스트리로부터 삭제하는 자격 또한 차단된다. 그러므로, 상기 시스템은 그러한 점에서 결합된 보안검사로 키와 값을 수정하고 삭제하는 방법을 포함한다.

도 2와 도 3은 본 발명의 일 실시예에 따른 실례가 되는 컴퓨터시스템(100)을 나타낸다. 레지스트리 침입 보호 드라이버(120)는 파일시스템 훅 드라이버(140)와 통신한다. 두 드라이버는 커널(링 0) 레벨(130)에 있다. 애플리케이션(160)은 상위 레벨(140)에서 작동한다. 애플리케이션(160)이 레지스트리(110)나 공유메모리(115)에 접근을 요청할 때, 훅 드라이버(140)와 연결된 보호 드라이버(120)는 상기 요청을 감시하고 처리한다.

패키지는 데이터를 운반하고 관계된 정보를 WIN32 프로세스와 같은 애플리케이션 프로그래밍 인터페이스의 구성요소인 명령센터에 제공한다. 통신드라이버는 애플리케이션 프로그래밍 인터페이스와 대다수의 디바이스 드라이버간의 통신을 처리한다. 상기 통신 드라이버는 디바이스 드라이버로 정보를 보내거나 디바이스 드라이버로부터 정보를 수정하기 위하여 애플리케이션 프로그래밍 인터페이스로부터 호출되는 디바이스 드라이버 입출력 제어기능의 싱글 셋을 제공한다. 상기 통신 드라이버는 명령센터에 프로세스가 패키지된 파일을 열기위한 시도를 하고 있다는 것을 알리기 위해서 훅 드라이버에 의해서 호출된다. 애플리케이션 프로그래밍 인터페이스와 함께 상기 디바이스 드라이버는.패키지된 내용을 정렬시키고 상기 내용에 접근을 유지하며, 창작자의 승인 선택을 제시한다. 상기 명령센터는 실행되는 패키지에 대해 감시할 수 있고 사용자에게 파일 이름을 패키지 페이로드(payload)에 저장할 것을 지시한다. 상기 명령센터는 패키지 페이로드가 상기 저장소에서 흡수된 파일 시스템 훅 드라이버에 알릴 수 있다. 그것은 애플리케이션이 패키지된 파일 열기를 시도하고 있는 것을 다이알로그 알림으로 사용자에게 알린다. 애플리케이션이 나갈 때, 또한 그것은 디바이스 드라이버(106)에 알린다. 상기 명령센터는 클립보드 접근을 막을 수 있고, 승인이 종료되었을 때 승인 디바이스 드라이버의 요청에서 애플리케이션을 종료시킬 수 있다. 승인 정보는 데이터베이스에 포함되고, 예를 들어 파일이름, 패키지아이디, 파일시스템 아이디, 파일 승인을 포함할 수 있다. 파일 승인은 제한되지는 않지만 파일이 오픈되는 시간이나 횟수, 파일이 더이상 오픈되지 않는 날짜, 프린트와 클립보드 승인을 포함할 수 있다.

파일시스템 훅 드라이버(140)는 패키지되거나 흡수된 파일에 접근하고자 하는 사용자로부터 초기화된 데이터 요청을 얻는다. 훅 드라이버(140)가 상기 요청을 받았을 때, 상기 프로세스에서 보안검사를 수행하고 그때 사용자에게 질문한다. 상기 프로세스는 그때 보호된 프로세스 리스트에 추가된다. 상기 레지스트리 감시자는 상기 프로세스가 보호되도록 알려져서 이후에 접근을 막을 수 있다.

도 4A-C는 본 발명의 일 실시예를 나타낸다. 본 발명에서 기술된 것은 키와 값에 접근을 막기위한 보안검사을 포함하는 레지스트리 감시시스템에서의 변화는 여기에 기술된 단계들과 같고, 즉 본 발명의 영역 내에 있다. 도 4A는 레지스트리 오픈 키호출에 대한 실례가 되는 필터링 연속을 나타낸다. 상기 호출은 호출 프로세스에서 레지스트리 키에 대한

핸들을 얻기위해 만들어 진다. 단계 304에서 프로세스 아이디와 레지스트리 키가 결정된다. 이 정보를 기초로 상기 프로세스가 보호된 프로세스 리스트를 검사하므로서 보호되는지가 단계 306에서 결정된다. 상기 보호된 프로세스 리스트는 호출이 초기화되는 거와 같이 계속적으로 업데이트된다. 상기 프로세스가 보호되면, 그때 단계 308에서 레지스트리 키가 거절 리스트상에 있는지를 결정한다. 상기 거절 리스트는 또한 계속적으로 업데이트된다. 상기 레지스트리 키가 거절 리스트에 있다면, 상기 프로세스는 단계 310에서 레지스트리 키로의 접근이 거부되고 상기 호출은 단계 312에서 성공적으로 필터된다. 상기 프로세스가 보호된 리스트에 있지 않거나 상기 레지스트리 키이름이 거절 리스트에 있지 않다면, 그때 단계 314에서 상기 요청은 완료되고 상기 호출은 단계 312에서 성공적으로 필터된다.

도 4B는 레지스트리키값 호출 필터링 연속에 대한 실례로서의 플로우차트이다. 상기 핸들에 대한 레지스트리 키값은 단계 316에서 요청된다. 상기 프로세스 아이디와 레지스트리 키값은 단계 318에서 결정된다. 단계 320에서 보호된 프로세스 리스트는 상기 프로세스가 보호되는지 결정하기 위하여 다시 정보를 요청하게 된다. 상기 프로세스가 보호된다면, 상기 레지스트리키는 거절 리스트상에 있는지가 단계 322에서 결정된다. 상기 레지스트리 키가 거절 리스트상에 있다면, 상기 프로세스는 단계 324에서 레지스트리 키값에의 접근이 거부되고, 상기 호출은 블록(326)에서 성공적으로 필터된다.상기 프로세스가 보호된 리스트상에 있지 않다면, 상기 요청은 단계 328에서 완료되고, 상기 호출은 블록(326)에서 성공적으로 필터된다. 상기 레지스트리키는 상기 거절 리스트상에 있지 않다면, 값 요청은 단계 330에서 처리되고 상기 값이 단계 332에서 거절 리스트상에 있는지를 결정한다. 상기 값이 거절 리스트 상에 있지 않다면, 상기 요청은 단계 328에서 완료되도록 허락되고, 상기 호출은 블록(326)에서 성공적으로 필터된다. 상기 값이 거절 리스트상에 없다면, 그때 단계(324)에서 접근은 상기 레지스트리키값에 거부되고, 상기 호출은 블록(326)에서 성공적으로 필터된다.

그때 핸들과 값은 삭제되거나 수정될 수 있다. 삭제 또는 수정 순서에 대한 전형적인 플로우차트가 도 4C에 도시된다. 삭제 또는 설정-값 호출은 단계 334에서 이루어진다. 상기 프로세스 아이디는 그때 단계 336에서 결정된다. 단계 338에서 상기 프로세스가 보호된 프로세스 리스트상에 있는지를 검사하므로서 상기 프로세스가 보호되는지가 결정된다. 상기 프로세스가 보호된 프로세스 리스트상에 있지 않다면, 상기 요청은 단계 340에서 완료되고 상기 호출은 레코드를 생성하고 상기 보호된 프로세스 리스트에서 레코드상에 들어오므로서 단계 342에서 지나가게 된다. 상기 프로세스가 보호된 프로세스 리스트에 있다면, 상기 요청은 단계 344에서 완료되는 것이 허락되지 않는다. 상기 호출은 레코드를 생성하고 상기 보호된 프로세스 리스트상의 레코드에 들어가므로서 단계 342에서 지나가게 된다.

상기 레지스트리가 본 명세서에서 기술된 방법에 따라 감시되는 점에서 더 나아가 나타낸 것은 레지스트리 감시시스템이다. 게다가, 본 발명의 일 실시예는 그러한 방법에 따라 레지스트리를 감시하기 위해 만들어진 컴퓨터를 포함한다. 본 발명의 실시예는 더 나아가 본 명세서에서 기술된 방법에 따라 레지스트리를 감시하는 프로그램으로 이루어진 기계를 읽을 수 있는 수단(machine-readable medium)을 포함한다.

다른 보호 메카니즘은 공유 메모리를 감시하고 관리하는 것이다. 공유 메모리는 둘 이상이 동작중인 작업이나 스레드간에 통신하기 위해 사용될 수 있다. 어떤 프로그램은 다른 프로세스가 접근할 수 있는 메모리부분을 생성한다.

공유 메모리는 또한 누설 데이터로 이용될 수 있다. 프로세스가 공유 메모리 위치에 정보를 기록하고 다른 프로세스가 그 위치에 상기 정보에 접근하면 데이터 누설은 일어날 수 있다.

업데이트된 공유 데이터만의 이용을 보증하기 위하여 연속중인 데이터를 프로세스가 접근하는 것을 막는 것은 공유 메모리를 닫는 것으로 알려져 있다. 공유 메모리 공간에 대한 접근은 첫번째 프로세스가 사용하는 동안 금지되고 그 후에 프로세스가 업데이트된 데이터에 공간 접근을 공유하는 것을 허락하기 위해서 락(lock)이 열린다. 본 기술분야에서 알려진 메모리 잠그기는 데이터 누출에 대한 해답이 아니다. 따라서, 데이터보안이 중요한 곳에는 공유 메모리로부터 데이터 누출을 제한하는 것이 필요하다.

본 명세서에서 기술된 실시예는 보호된 데이터에 접근하는 것으로부터 공유 페이지를 가진 프로세스를 금지한다. 페이지를 공유하는 것으로부터의 블록킹 프로세스는 공유 메모리로부터 데이터 누출을 줄이거나 제거할 수 있다. 실시예는 프로세스가 보호된 데이터파일을 여는 것을 허가받기 전에 보안검사를 포함한다. 상기 보안검사는 상기 프로세스의 공유 메모리 상태 조사와 상기 프로세스가 보호된 프로세스 리스트상에 있는지를 결정하는 것으로 이루어진다.

상기 공유 메모리 블록킹 프로세스는 두 가지 중요한 부분으로 나뉠 수 있다. 첫번째 부분은 데이터를 수집하기 위해 사용되는 호출을 보존, 보류, 삭제를 포함하는 훅킹 서비스 호출(hooking service calls)로 이루어진다. 두번째 부분은 보호된 데이터에 대한 접근을 제어하기위한 데이터 이용을 포함한다.

도 2와 도 3에서, 훅 드라이버(202)가 요청을 받을 때, 상기 훅 드라이버(202)는 상기 프로세스가 보호되고 공유 페이지가 있는지를 결정하는 레지스트리 침입 보호 드라이버(204)를 호출한다. 보호 드라이버(204)는 보호된 프로세스와 고유 메모리 레코드를 검사한다. 공유가 없으면 요청이 승인될 수 있고, 공유가 있으면 보호 드라이버(204)는 상기 프로세스가 보호되지 않는 훅 드라이버(202)에 알린다. 그리고 그 결과로 상기 요청은 거절된다. 상기 페이지가 그때 공유되지 않으면 사용자는 보호된 데이터에 접근하도록 허락되고 상기 프로세스는 보호된 프로세스 리스트상에 놓이게 된다.

프로세스들이 보호되는지 그리고 페이지가 공유되는지를 결정하는 보안검사를 포함하는 공유 메모리 블록킹의 변화는 본 명세서에서 기술된 상기 단계들과 동등하고, 그러므로 본 발명의 사상과 범위 내에 있으며, 본 기술분야의 종사자는 이해할 것이다. 공유 메모리 블록킹의 실시예는 도5A-D에 도시된다. 도 5A는 전형적인 페이지 저장 호출 필터링 순서의 플로우차트이다. 상기 저장 호출은 요청하는 애플리케이션에 대하여 메모리의 페이지를 저장한다. 상기 필터링 순서는 단계 402에서 상기 호출을 제공하므로서 시작된다. 단계 404에서 요청 패러미터에 기초를 두고, 상기 호출은 그때 상기 페이지가 공유될 수 있는지를 처음 결정하므로서 필터된다. 상기 페이지가 공유될 수 없다면 상기 요청은 단계 406에서 완료되도록 허가된다. 상기 페이지가 공유될 수 있으면, 레코드를 생성하고 상기 레코드를 공유 메모리 리스트에 등록시키므로서 상기 저장 호출은 단계 408에서 통과하게 된다. 상기 레코드는 프로세스 아이디, 페이지 번호, 공유 회수를 포함한다. 도 5B는 페이지 위탁 호출을 필터링하는데 대한 전형적인 순서를 나타낸다. 상기 페이지 위탁 호출은 상기 요청 프로세스나 단계 410에서 연속적인 요청 프로세스에 대한 메모리 페이지를 위탁하기 위하여 제공된다. 상기 페이지가 다른 프로세스에 의해 공유되면, 공유 메모리 접근으로 단계 412에서 결정된다. 상기 페이지가 공유되면, 보호된 프로세스 리스트에 접근하므로서 단계 414에서 어떤 공유 프로세스가 보호되는지가 결정된다. 사용자로터 보호된 파일을 열기위한 시도가 있을 때 상기 보호된 프로세스 리스트는 생성된 레코드를 연속적으로 컴파일하므로서 생성된다. 어느 프로세스라도 보호된다면, 페이지 공유는 단계 416에 도시된 바와 같이 허락되지 않는다. 그리고, 상기 저장 호출은 단계 418에서 통과하게 된다. 두 프로세스가 보호되지 않는다면, 새로운 공유 메모리 레코드는 단계 420에서 생성되고, 상기 공유 메모리 리스트는 단계 422에서 새로운 레코드에 포함된 정보로 업데이트된다. 상기 공유 횟수는 또한 상기 페이지를 공유하는 프로세스에 대하여 업데이트된다. 상기 페이지가 공유되지 않으면, 상기 저장 요청은 단계 424에서 완료된다. 저장된 애플리케이션은 그때 수행된다. 도 5C는 페이지 자유 호출을 위한 필터링 순서의 실례를 나타낸다. 상기 페이지 자유 호출은 약간 또는 모든 어드레스 공간의 메모리 페이지를 제거하기 위해 사용될 수 있다. 상기 호출은 단계 426에서 제공된다. 단계 428에서 상기 프로세스가 보호된 프로세스 리스트를 검사하므로서 보호되는지가 결정된다. 상기 프로세스가 보호되면, 상기 페이지는 단계 430에서 보호된 데이터를 상제하기 위하여 덮어쓰게 되고, 상기 덮어쓴 페이지와 같은 페이지 번호의 공유 메모리 리스트의 모든 레코드는 단계 432에서 삭제된다. 상기 프로세스가 보호되지 않는다면, 상기 보호되지 않는 프로세스 페이지에 상응하는 페이지 번호의 공유 메모리 리스트로부터의 모든 레코드는 또한 단계 432에서 삭제된다. 일단 페이지가 삭제되고, 상기 페이지 자유 호출은 단계 434에서 통과하게 된다.

프로세스가 훅 드라이버에 의하여 패키지된 데이터를 열도록 허락되기 전, 보안검사는 상기 프로세스의 공유 메모리 상

태를 결정하기 위하여 실행된다. 즉 상기 프로세스에 대한 접근이 승인될 수 있는지를 결정하기 위하여 상기 보안검사가 실행된다. 공유 페이지를 가진 어떤 프로세스도 패키지된 데이터에 접근하는 것은 허락되지 않는다. 도 5D는 공유 메모리 상태 보안검사의 플로우차트이다. 단계 490에서 상기 보안검사는 초기화된다. 단계 492에서 상기 프로세스가 0보다 큰 할당 수의 페이지를 가지는지가 결정된다. 상기 프로세스가 0이상의 할당 수의 페이지를 가진다면, 상기 프로세스는 보안검사를 실패하고 접근은 단계 414에서 상기 프로세스에 거절된다. 상기 프로세스가 0이상의 할당 수의 공유 페이지를 가지지 않는다면 그때 단계 416에서 상기 보호된 데이터에로의 접근은 승인되고 상기 프로세스는 상기 보호된 프로세스 리스트에 추가된다.

시스템은 본 명세서에서 제공된 방법에 따라 메모리를 차단다는 점에서, 더 나은 실시예는 공유 메모리 블록킹 시스템으로 향하게 된다. 실례로서 공유 메모리 블록킹 시스템은 보호된 저장소에서 하나 이상의 패키지된 파일을 정렬시키고 파일 내용 접근을 유지하기 위한 애플리케이션 프로그래밍 인터페이스로 이루어진다. 상기 애플리케이션 프로그래밍 인터페이스는 패키지 접근을 감시하기 위하여 명령센터를 포함한다. 파일시스템 훅 드라이버는 명령센터와 통신하고 공유 메모리 블록킹을 외부로 전송하기 위하여 레지스트리 침입 보호 드라이버와 통신한다.

리시버가 공유 메모리 블록킹 시스템을 포함한다는 점에서, 다음에 기술된 것은 발송자에 의해 제공된 보호 파일내용에 접근하기 위한 리시버 구성요소를 가지는 보호 데이터 전송시스템이다.

본 발명에 따른 다른 보호 메카니즘은 데이터의 백-채널링이며 본 명세서에서 기술된다.

권한이 없는 접근으로부터 데이터를 보호하기 위하여, 상기 데이터는 암호화될 수 있다. 암호화 알고리즘은 통상 한 쌍의 키(암호화키 하나와 복호화키 하나)를 이용하여 설계된다. 암호화는 정보를 암호화하기 위해 사용되고 그것은 암호된 파일로서 보내진다. 해커가 교환되는 데이터를 방해할 수 없으므로, 암호화는 또한 두 컴퓨터간의 보안 연결을 위해 사용될 수 있다.

암호화 컨테이너는 컴퓨터시스템에서 데이터 보호를 위해 사용될 수 있다. 암호화 컨네이너는 보호 파일이 복사되고 저장되는 디스크에 저장영역이 있다. 이것은 암호화되고 파일시스템으로서 설치되고 작동되는 파일을 생성시키므로서 실행될 수 있다. 종래기술의 암호화 컨테이너는 내용의 보안을 위태롭게하지 않고 내용 배포자가 내용을 배포하도록 하는 것을 허락한다. 클라이언트측 컨테이너-오프너(container-opener) 애플리케이션은 암호화 컨테이너에 접근하기 위해 사용된다. 상기 클라이언트측 컨테이너-오프너는 몇 가지 방법으로 암호화 컨테이너의 데이터에 대한 접근을 제한할 수 있다. 예를 들어, 데이터가 어떤 시간에만 접근되거나 지불키의 증명으로만 접근될 수 있다.

일단 상기 컨테이너가 열리고 상기 데이터가 해제되면 종래기술의 암호화 컨테이너 시스템은 제한된 보안을 포함한다. 종래기술의 소프트웨어와 운영체제에서 작동되는 컴퓨터상에서 파일이 열려질 때, 상기 데이터는 클립보드나 다른 시스템 애플리케이션과 같은 애플리케이션에 누출될 수 있고 복사되거나 프린터와 같은 시스템 동작동안 불안정하게 될 수 있다. 이는 시스템을 손상시킨다.

어떤 클라이언트측 컨테이너-오프너는 사용자가 보호데이터를 볼 수 있도록 하는 통합 데이터-출력 메카니즘으로 이루어질 수 있다. 예를 들어, 폴리오(Folio: NextPage,Inc.)는 파일을 출력하는 뷰어 브라우저를 제공하고 상기 뷰어 브라우저는 애플리케이션 레벨 제어를 차단하므로서 데이터에 대한 권한없는 사용을 막는다. 그러나, 이는 폴리오에 있는 문서가 출력될 때 데이터가 클립보드에 저장될 수 있고 상기 시스템 레벨로부터 다른 방법으로 공격받을 수 있음을 의미한다.

상기 클라이언트측 컨테이너-오프너 애플리케이션에 의해 투명무늬를 넣거나(watermarking) 디지털 지문채취를 하는 것(fingerprinting)은 열려지고 암호화된 컨테이너로부터 삭제된 내용의 출처를 추적하는 데 사용될 수 있다. 이것은 권한없는 방법으로 배포된 내용의 흔적을 찾아내도록 할 것이다. 그러나, 이것은 여전히 상기 데이터가 권한없는 사용자에 의해 노출되는 것도 허가한다.

나타낸 실시예는 메모리 저장시스템을 포함하는 컴퓨터시스템이 이러한 애플리케이션이나 보안위반의 변형없이 불확실한 애플리케이션으로 상기 저장소로부터 보호 데이터의 사용을 포함하도록 허락한다. 이것은 이러한 불확실한 애플리케이션(다른 말로, 이런 종류의 데이터의 분리 그리고 시스템 메모리에서보다 상기 저장소에서의 이런 데이터의 생성)에 의해 생산되거나 이용되는 데이터의 백-채널링을 실행시키므로서 일어난다.

도 6을 참조하여, 바람직한 실시예로, 백-채널링 방법은 컴퓨터시스템(500)에서 실행된다. 컴퓨터시스템(500)은 메모리(510)를 포함하며 상기 메모리(510)는 몇 가지 방법으로 배열되고 여러 메모리 시스템과 메모리 매체 종류를 포함할 수 있다. 메로리(510)상의 상주는 적어도 하나의 저장소(520)에 있다. 바람직한 실시예로, 저장소(520)는 파일의 논리구조를 이용하고 단일물로 내장된 것내에서, 강한 파일구조는 이종의 파일 소스로부터 데이터를 처리할 수 있다. 메모리(510)와 하드웨어는 커널(링 0) 레벨(530)에서 커널-모드 애플리케이션 상주에 의해서만 직접적으로 접근될 수 있다. 그러한 커널-모드 애플리케이션은 상기 파일시스템 보안드라이버(540)이다. 상위 레벨(도 6에서 총괄하여 도시된 550)에 있는 상위-레벨 애플리케이션(560)은 화살표로 도시된 바와 같이 상기 커널(링 0) 레벨(530)에서 애플리케이션 상주에 의해서만 메모리에 접근한다. 본 발명의 바람직한 실시예에서, 파일시스템 보안 드라이버(540)와 관계된 상기 보안시스템은 상위-레벨 애플리케이션(560)에 의한 데이터 요청이 파일시스템 드라이버(540)에 의해 항상 처리되는 것을 보증한다. 상기 파일시스템 보안 드라이버(540)는 보호 프로세스 작동에서 정보를 유지하고, 파일은 상위-레벨 애플리케이션이 보호 파일과 다른 저장 파일을 참조하기 위해 이용하도록 처리하고, 저장 파일은 파일시스템 보안 드라이버(540)가 보호 파일과 다른 저장 파일에 접근할 수 있도록 처리한다. 상기 파일시스템 보안 드라이버(540)는 저장 정보에 대한 접근이 제한없이 가능하도록 보증하기위하여 이러한 정보를 이용한다. 파일 열기, 파일 읽기/쓰기, 파일 정보, 파일 변환 요청을 방해하고 작동시키므로서 이러한 것이 이루어진다.

도 7을 참조하여, 프로세스는 파일시스템 보안 드라이버(540)에 의해 수신된 파일 열기 요청을 보낸다. 파일 열기 요청(600)을 받으므로서, 보호 파일(605)이 있다면 상기 요청은 보는 것이 실행된다. 그렇다면, 그때 상기 요청은 상기 요청 프로세스가 보호 프로세스 리스트(610)상에 있는지를 확인하도록 조사된다.

상기 요청이 보호 파일에 대한 것이고 상기 요청 프로세스가 상기 보호 프로세스 리스트상에 있지 않다면, 그때 보안검사는 상기 요청 프로세스(615)에서 보안 드라이버에 의해 실행된다. 상기 프로세스가 보안검사를 통과하지 못한다면, 그때 상기 요청 프로세스는 요청된 파일(620) 대신 0-바이트 태그파일에 접근하게 된다. 상기 프로세스가 상기 보안검사를 통과하면, 그때 승인은 검사되고, 사용자는 사용자가 보호 파일(625)을 열고자하는지에 대해 질문을 받는다. 승인이 허락되지 않거나 사용자가 그것을 열고자하지않는다면, 그때 상기 요청 프로세스는 요청된 파일(620) 대신 0-바이트 태그파일에 접근하게 된다. 사용자가 그것을 열고자 한다면, 상기 프로세스는 보호 프로세스 리스트(630)에 추가된다. (보호 파일의 열기에 대한 정보는 허가된 파일 열기 횟수를 검사하기 위하여 보안 드라이버부를 통과시키게 될 것이다.) 상기 프로세스는 파일 처리로 승인되고 상응하는 저장 파일 처리가 생성될 것이다.; 이러한 처리는 열린 파일 리스트(635)에 색인으로 저장될 것이다. 이러한 방법으로 파일은 열리고 백-채널링(640)에 대해 준비된다.

상기 요청이 보호 파일에 대한 것이고, 상기 요청 프로세스가 보초 프로세스 리스트상에 있고 그때 상기 프로세스가 이전에 상기 요청 파일을 열지 않았다면, 상기 검사와 질문(225)은 발생될 것이다. 상기 프로세스는 전에 그것을 열었다면, 상기 프로세스 작동은 백 채널된다. 상기 프로세스는 파일 처리로 승인되고 상응하는 저장 파일 처리는 생성될 것이다.; 이러한 처리는 열린 파일 리스트(635)에 색인으로 저장될 것이다. 파일은 열리고 백-채널링(640)에 대해 준비된다.

상기 요청이 보호 파일에 대한 것이 아니라면, 그때 상기 보호 프로세스 리스트는 상기 요청이 보호 프로세스(645)에 의해 만들어졌는지 아닌지를 보기위해 의견을 묻게된다. 그렇다면, 그때 검사는 상기 파일이 있는지를(650) 확인하게 된다. 그렇지 않다면, 그때 백-채널을 위하여, 파일은 상기 파일 요청(655)과 상응하는 상기 저장소에 생성된다. 이러한 방법으로, 보호 프로세스는 상기 저장소의 외부에 파일(불안정한 파일)을 생성하는 것을 허가하지 않는다. 이것이 백-채널링이고 보호 프로세스가 상기 저장소에 의해 보호되지 않는 파일에 보호 정보를 " 누출" 하는 것을 승인하지 않는다.

상기 요청은 보호 프로세스에 의한 것이고, 안전하지 않은 파일에 대해서는 제외하며, 그때 상기 파일 열기 요청에 대한 상기 파일 요청 플래그(flag)는 상기 불안전한 파일에서 어떤 데이터도 삭제될 수 없거나 수정될 수 없도록 변경된다.

이러한 방법으로 보호 프로세스는 상기 저장소의 외부에서 비-보호 데이터에 접근할 수 있으나, 그것에 쓸 수는 없다. - 보호 데이터가 불안전한(저장되지 않는) 파일에 씌여지는 것을 막는다. 상기 변경된 요청은 완성(665)을 위한 상기 파일시스템을 통과하게 된다.

상기 요청이 보호 파일에 대한 것이 아니고 보호 프로세스에 의해 일어나지 않는다면, 그때 상기 요청은 보호 관계가 없고, 상기 요청은 완료(665)로부터 파일시스템을 통과하게 된다.

이러한 방법으로, 보호 데이터는 보호 프로세스에 의해서만 접근될 수 있고, 보호 프로세스에 의해 열려진 새 파일은 상기 저장소에서 항상 생성되고, 보호 프로세스에 의해 열려진 비보호 파일은 씌여질 수 없다.

도 8에 도시된 바와 같이, 파일 읽기/쓰기 요청이 파일시스템 보호 드라이버(540)(도 6 참조)에 의해 차단될 때 백-채널링은 또한 실행된다. 그러한 요청이 수신될 때(700), 검사는 상기 파일 처리가 열려진 파일 리스트상에 있는지를 확인하도록 실행된다(705). 그렇다면, 그때 상기 파일은 상기 저장소에 있다. 검사는 상기 요청 프로세스가 상기 보호 프로세스 리스트상에 있는지를 확인하기위해 실행된다(710). 그렇다면, 그때 상기 요청은 상기 상응하는 저장 파일, 즉 읽기/쓰기 요청에 대한 백-채널링(720),에서 상기 요청을 수행하므로서 완성된다(715). 그렇지 않다면, 그때 상기 요청은 비보호 프로세스에 의해 보호 파일에 대한 것이고, 상기 요청은 거부된다(725).

상기 파일 처리가 검사에서 상기 열려진 파일 리스트상에 있지않다면, 다음 검사는 상기 프로세스가 보호 프로세스 리스트상에 있는지를 확인하게 된다(730). 그렇지 않다면, 그때 상기 파일시스템은 상기 요청을 완료하도록 허락된다(735). 그렇다면, 그때 상기 요청은 그것이 쓰기 요청이 인지를 확인하기 위해 실행된다(740). 그렇지 않다면, 상기 파일은 열려질 수 있다. 그것이 비보호되고 상기 프로세스가 보호됨에도 불구하고 어떤 보호 데이터도 비보호 파일을 읽으므로서 누출될 수 없기 때문에 상기 파일은 열려질 수 있는 것이다. 상기 요청은 파일시스템을 통과하게 된다(735). 상기 파일이 비보호된다면, 상기 프로세스는 보호되고, 상기 요청은 쓰기이다. - 이러한 상황에는 데이터 누출에 대한 위험이 있다. 그러므로 상기 쓰기 요청은 차단된다(725).

파일정보 요청은 다르게 처리된다 - 정확한 파일크기는 비보호 프로세스에 의해서만 접근될 수 있다. 이는 파일크기를 속이는 것이다 - 바람직한 실시예에서, 상기 파일시스템은 0-바이트 파일에 연결된 상기 보호 파일의 참조를 확인한다. 그리고 파일크기 요청이 상기 파일시스템에 의해 처리된다면 제로 크기를 리턴한다. 바람직한 실시예에서, 비-보호 프로세스는 보호 파일크기를 확인하는 것이 허락된다. 그러므로, 도 9를 참조하여, 상기 요청이 수신될 때(900), 검사는 그것이 저장 파일인지를 확인하기 위하여 실행된다(805). 그것이 저장 파일이 아니라면, 그때 상기 요청은 완료하기위해 상기 파일시스템을 통과하게 된다(810). 그것이 저장 파일이나 저장 파일의 이름에 대한 처리라면, 그때 상응하는 실제의 저장 파일의 크기는 검사되고 리턴된다(815).

파일변환 요청 - 읽기, 쓰기, 열기가 아닌 변환 요청 - 은 파일 쓰기와 유사하게 처리된다. 도 10을 참조하면, 파일변환 요청이 수신될 때(900), 상기 요청은 참조된 파일이 저장 파일인지를 확인하기 위하여 실행된다(905). 그렇다면, 그때 상기 열린 파일 리스트와 다른 데이터베이스는 새 파일정보로 업데이트된다(910). 상기 파일이 저장 파일이 아니라면, 그때 상기 프로세스는 그것이 보호 프로세스인지를 확인하기 위하여 검사된다(915). 그것이 보호 프로세스가 아니라면, 그때 어떤 보호 관계도 관련되지 않고, 상기 요청은 완료에 대해 상기 파일시스템을 통과하게 된다(920). 그것이 파일이 보호되지 않는 것을 제외한 보호 프로세스이라면, 보호 프로세스가 존재하는 데이터를 이전에 손상시키지 않을 것을 보증하기 위하여 그때 상기 요청은 차단된다(925).

상기 드라이버가 본 명세서에서 제공된 방법에 따른 백-채널링을 가진 저장 시스템을 실행시킨다는 점에서, 본 발명의 다음 실시예는 파일시스템 보안 드라이버로 향하게 된다.

실례의 파일시스템 보안 드라이버는 커널 레벨에서 드라이버 상주로 이루어지며, 상기 커널 레벨은 파일시스템 요청을 감시하고 필요시 저장 파일의 저장과 생성에서 파일 상주에 제한된 접근을 허가한다.

더 나아가서 나타낸 것은 발송자에 의해 제공된 보호 파일 내용에 접근하기 위한 리시버 구성요소로 이루어지는 보호 데이터 전송시스템이다. 상기 리시버는 본 명세서에서 제공된 방법에 따라 작동되는 저장시스템과 파일시스템 보안 드라이버를 포함한다.

여전히 더 나아가서 나타낸 것은 저장시스템과 파일시스템 요청을 감시하기 위하여 프로그램된 컴퓨터-읽기 가능한 수단을 포함하기 위하여 형성된 컴퓨터이고 본 명세서에서 제공된 방법에 따라 저장 파일의 저장과 생성에 있어 파일 상주에 제한된 접근을 허가한다.

본 발명에 따른 다른 보호 메카니즘은 본 명세서에서 상세히 기술하게될 스택 위치정하기(stack positioning)이다.

일반적으로 컴퓨터시스템은 하나 이상의 지역 또는 네트워크 데이터 저장장치를 포함하고 있다. 그러한 컴퓨터시스템상에서 실행되는 일반적인 응용프로그램은 운영체제에 의해 제공되는 표준파일시스템 서비스를 호출하므로서 그런 데이터 저장장치에 접근하며, 상기 표준파일시스템 서비스로는 상기 데이터 저장장치에서 파일 생성, 파일 읽기, 파일 쓰기에 대한 서비스가 있다.

디바이스 드라이버는 일반적인 입출력 작동의 디바이스 특유의 측면을 실행시키는 컴퓨터 기반의 명령어 세트이다. 일반적인 운영체제에서, 디바이스 드라이버와 같은 소프트웨어 애플리케이션은 " 커널모드" 나 " 사용자모드" 에서 작동된다. 가상 디바이스 드라이버에는 커널모드에서 작동되는 거와 같은 운영체제 커널에 직접적인 접근을 하는 디바이스 드라이버 타입이 있다. " 커널모드" 는 프로세서의 아주 특별한 메모리 접근모드이다. " 사용자모드" 는 프로세서의 특별하지 않은 메모리 접근모드이다. 상기 메모리 접근모드는 상기 프로세서의 하드웨어상태에 대한 부분이다. 커널모드 접근은 가상 디바이스 드라이버가 매우 낮은 레벨에서 시스템과 하드웨어 리소스사이에 상호작용을 하도록 허락한다. 상기 커널모드 특별 레벨은 또한 " 링 0" 로 알려져 있고, 사용자모드 특별 레벨은 또한 " 링 3" 로 알려져 있다. 커널모드 접근은 상기 가상 디바이스 드라이버가 매우 낮은 레벨에서 시스템과 하드웨어 리소스간의 상호작용을 허락한다.

종래의 운영체제에서, 디바이스 드라이버는 서로서로 층층으로 이루어져 설명될 수 있다. 상기 층구조는 또한 때때로 스택이나 연속 호출로 언급된다. 일반적으로 하드웨어 디바이스를 제어하는 것은 가장 낮은 레벨 디바이스 드라이버이다. 하드웨어 디바이스 위에 하나의 디바이스 드라이버만 있다면, 상기 드라이버는 단일 드라이버로 호출된다. 그러나, 대다수의 드라이버는 가장 낮은 레벨 드라이버상에 위치될 수 있다. 가장 낮은-레벨 드라이버에 의해 제어되는 디바이스나 하드웨어 디바이스에 대한 입력과 출력 요청(" 입출력 요청")은 처음엔 가장 높은-레벨 드라이버, 잇달아 저-레벨 중간 드라이버, 그리고 마지막으로 가장 낮은-레벨 드라이버에 의해 처리된다.

파일시스템 드라이버는 일반적으로 하드디스크 드라이브와 같은 데이터 저장장치에 대한 디바이스 드라이버 위에 층층이 있는 최상위-레벨 드라이버이다. 상기 파일시스템 드라이버는 상기 파일시스템에 명령하는 입출력 요청(예를 들어, 파일과 디렉토리를 생성, 열기, 확장, 삭제하라는 요청)의 고-레벨 측을 실행시킨다. 다수의 파일시스템 드라이버는 단일의 컴퓨터상에 있고, 파일시스템 드라이버는 파일시스템의 다른 타입(예를 들어, FAT와 NTFS 파일시스템)에 명확해질 수 있다.

이는 설치가능한 파일시스템 관리자와 층을 이룬 디바이스 드라이버로 이루어진 운영체제에서 파일 입출력 요청을 감시하는 기술분야로 알려져 있다. 상기 운영체제는 레드몬드(Redmond), 워싱턴(Washington)의 마이크로소프트사로부터 입수할 수 있는 윈도우즈 95, 운도우즈 98 그리고 윈도우즈 Me가 있으며 본 명세서에서는 총괄적으로 " 윈도우즈 9x" 라 한다. 윈도우즈 9x 운영체제에서, 파일시스템 감시는 설치가능한 파일시스템 관리자를 가진 파일시스템 응용 프로그램 인터페이스 훅(hook)을 등록하므로서 완성될 수 있다. 윈도우즈 9x는 파일시스템에 대한 입출력 요청을 감시하기 위해 사용되도록 설계된 IFSMGR_InstallFileSystemApiHook라는 기능을 제공한다. 이러한 서비스는 가상 디바이스 드라이버가 상기 파일시스템 호출에서 훅킹(hooking)하므로서 모든 파일시스템 작동을 감시하는 것을 허가한다. IFSMGR_InstallFileSystemApiHook에서 시스템 초기화동안의 호출에 의하여, 가상 디바이스 드라이버는 모든 파일시스템 요청의 스택에 스스로 삽입된다.

약간의 다른 접근은 객체지향 운영체제에서 파일시스템을 감시하기위해 사용되어 왔다. 상기 객체지향 운영체제는 레이몬드, 워싱턴의 마이크로소프트사로부터 입수할 수 있는 윈도우즈 NT 운영체제와 후속의 윈도우즈 2000 운영체제가 있으며 본 명세서에서는 총괄적으로 " 윈도우즈 NT" 라 한다. 윈도우즈 NT에서 입출력 요청은 소프트웨어 애플리케이션과 드라이버간의 통신을 위해 이용되는 입출력 요청 패킷으로 알려진 데이터 구조로 설명된다. 하드웨어 디바이스에 있는 모든 입출력 요청 패킷은 커널모드에서 작동되는 디바이스 드라이버에 의해 처리된다. 고-레벨, 중간, 그리고 저-레벨 드라이버는 주어진 입출력 요청을 완료시키기 위하여 입출력 요청 패킷을 교환한다. 최저-레벨 드라이버는 하드웨어의 직접적인 제어를 위해 하드웨어 접근 층(Hardware Access Layer: HAL)로 알려진 NT 층을 호출한다.

그것은 파일시스템 감시자를 디바이스 드라이버 오브젝트로서 실행시키기 위한 윈도우즈 NT 시스템으로 알려져 있다. 상기 디바이스 드라이버 오브젝트는 필터 디바이스 오브젝트를 생성하고 타겟 파일시스템 디바이스 오브젝트에 그 오브젝트에 붙인다. 그래서 상기 파일시스템 감시자는 상기 감시되는 데이터 저장장치에 명령을 내리는 모든 입출력 요청 패킷을 확인할 것이다.

권한이 없는 작동을 막기위해 컴퓨터의 파일시스템부에서 필요한 것이 있다. 제한없이, 권한이 없는 작동은 보호 파일이나 보호 파일시스템부터 비보호 파일시스템 디바이스 드라이버까지의 데이터 해제와, 컴퓨터 바이러스와 같은 부당하고 권한이 없으며 우연한 데이터 변경 또는 데이터 손상을 포함한다. 상기 타겟이 된 데이터 저장장치 위의 최 상위-레벨 디바이스 드라이버보다 더 높은 레벨에서, 파일시스템에서 작동을 감지하고 검사하기위해 사용되는 파일시스템 감시자와 같은 종래의 파일시스템 드라이버는 소프트웨어 애플리케이션이나 층을 이룬 디바이스 드라이버 또는 자체적으로 층을 이룬 시도에 의해 권한없는 작동을 막기 위한 효과적인 보안측정을 일반적으로 포함하지 않았다.

본 명세서상에서 기술된 " 디바이스 드라이버" 나 " 드라이버" 는 직접적으로 또는 간접적으로 하드웨어 디바이스에 접근하고 디바이스를 제어하는 컴퓨터 기반 명령어를 포함한다는 것은 이해된다. 상기 디바이스는 제한없이 디바이스 드라이버, 가상 디바이스 드라이버(virtual device drivers: VxDs), 커널모드 구조를 이용한 명령어, WIN32 드라이버 모델(WIN32 driver model: WDM)을 이용한 명령어, 컴퓨터 언어에 있어서 컴퓨터, 컴퓨터 구조, 네트워크 또는 운영체제로 명령을 내리는 다른 명령어를 포함한다.

도면에 도시된 실시예는 " 파일시스템 감시자" 로 도시된 목적으로 설명된 디바이스 드라이버를 구성함에도 불구하고, 본 명세서에서 기술된 " 파일시스템 감시자" 라는 용어는 일반적으로 본 발명의 스택 위치시키기(stack positioning)를 이용한 종류의 디바이스 드라이버를 말한다. 본 발명의 범위내에 있는 디바이스 드라이버는 디바이스 드라이버에 의해 수행될 수 있는 일종의 유용한 기능을 수행할 수 있다. 상기 디바이스 드라이버는 제한없이 일반적인-목적의 감시, 승인 감시, 필터링, 암호화, 복호화, 바이러스 감지, 데이터 미러링, 어떤 디바이스에 대한 입출력기능, 그리고 또 다른 기능을 포함하며, 감시하는 거나 파일시스템에 관계된 기능에 대해서는 제한되지 않는다. 스택 포지셔닝(positioning)을 이루는 디바이스 드라이버는 본 발명의 적당한 범위내에 있다.

본 발명의 일 실시예는 윈도우즈 9x 운영체제를 기반으로 한다. 도 11을 참조하면, 윈도우즈 9x 운영체제의 구성은 시스템 보호의 다른 레벨을 제공하는 사용자모드 코드(1000)와 커널모드 코드(1030)로 나뉜다. 일 실시예로, 상기 사용자모드 코드(1000)는 16-비트와 32-비트 소프트웨어 애플리케이션(1021-1022), 그리고 대다수의 MS-DOS 가상머신(1025)을 작동가능하도록 하는 시스템 가상머신(1020)을 포함한다.

이러한 실시예에서, 상기 커널모드 코드(1030)는 가상머신 매니저(1040), 본 발명의 파일시스템 감시자(1050), 설치가능한 파일시스템 매니저(1060)과 같은 저-레벨 운영체제 서비스와 가상 디바이스 드라이버로 이루어진다.

상기 설치가능한 파일시스템 매니저(1060) 아래는 FAT와 NTFS와 같은 파일시스템에 대한 대다수의 파일시스템 드라이버(1070-1072)이다. 상기 파일시스템 드라이버(1070-1072) 아래는 블록 입출력 서브시스템(1080)이다. 상기 블록 입출력 서브시스템(1080)은 파일시스템체계, 입출력 포트에 대한 단일의 드라이버(1082)와, 층을 이룬 대다수의 디바이스 드라이버(1083-1084)를 통과하는데 대한 요청을 관리하는 입출력 슈퍼바이저(supervisor)(1081)를 포함한다.

이러한 실시예에서, 첫번째 디바이스 드라이버(1050)는 사용자모드 코드(10)와 사용자모드(1000)에서 작동하는 애플리케이션(1021-1022)로부터 모든 입출력 요청을 차단하며, 상기 입출력 요청은 상기 설치가능한 파일시스템 관리자(1060)에 보내진다. 상기 첫번째 디바이스 드라이버(1050)은 감시가 가능하고, 요구된다면, 상기 설치가능한 파일시스템 관리자(1060), 파일시스템 드라이버(1070-1072)와, 블록 입출력 서브시스템(1080)에서 일어나는 모든 파일시스템 작동을 여과하여 제거할 수 있다.

IFSMGR_InstallFileSystemApiHook의 시스템 초기화 동안 호출에 의하여, 상기 첫번째 드라이버(1050)는 모든 파일시스템 요청의 스택에서 기능적으로 최상의 위치에서 삽입되는 시간에서 운영체제가 시작되거나 재시작되었을 때 그러한 호출로 잠겨진다. 층을 이룬 다수(1083-1084)에서 각 드라이버 아래의 상기 설치가능한 파일시스템 관리자(1060)로부터, 입출력 요청은 최상 레벨부터 최저 레벨까지 통과된다. 그리고 상기 디바이스는 또한 상기 입출력 요청의 소스에서 스택을 뒤로 통과시키는 요청에 대한 결과를 관찰할 수 있다. 스택에서의 각 디바이스 드라이버는 자체적으로 입출력을 서비스할 수 있고, 저 레벨로는 상기 입출력 요청을 통과시키지 않고, 또는 요구된다면, 자체적으로 새로운 입출력 요청을 발생시킨다. 그런 디바이스 드라이버는 인터럽트나 이용할 수 있도록 하는 디바이스와 같은 대기시간을 요청하는 기능을 실행시킬 수 있다. 그런 대기시간동안 상기 디바이스 드라이버는 단순히 그것의 호출자(caller)를 리턴하여, 호출 애플리케이션이나 디바이스 드라이버가 상기 입출력 요청과 병렬로 다른 작업을 수행하는 것을 허락한다. 양자택일로, 상기 호출 애플리케이션이나 디바이스 드라이버는 상기 입출력 요청이 완료될 때까지 단순히 기다릴 수 있다(" block" : 블록).

또 다른 실시예에서, 도 12를 참조하여 설명한 바와 같이, 본 발명은 윈도우즈 NT 운영체제를 기반으로 될 수 있다. 본 기술분야에서 알려진 바와 같이, 윈도우즈 NT 상에서 사용자모드로 작동되는 애플리케이션(1100)은 입출력 요청을 운영체제 서비스(1110)에 보낼 수 있다. 입출력 관리자(1120)는 입출력 요청을 받고, 여러 드라이버 사이의 입출력 요청 패킷의 이동을 조정한다. 양자택일로, 상기 여러 드라이버는 상기 여러 드라이버 사이의 정보 이동을 조정하기 위하여 입출력 관리자(1120)나 다른 디바이스를 이용하지 않고 서로 직접적으로 통신할 수 있다.

윈도우즈 NT와 같은 운영체제의 종래의 입출력 시스템은 입출력 요청을 처리하기위한 다수의 디바이스 드라이버(1130-1132)로 이루어진다. 예를 들어, 그러한 디바이스 드라이버는 파일시스템 드라이버(1130)와, 층을 이룬 다수의 디바이스 드라이버(1131-1132)로 설명된다. 일반적으로 상기 입출력 관리자(1120)는 입출력 패킷을 상기 입출력 요청의 타겟을 관리하기 위한 능력이 있는 상기 파일시스템 드라이버(1130)로 전달한다. 그러나, 본 기술분야에 알려진 바와 같이, 파일시스템 감시자(1150)는 객체지향 방식에서 다른 디바이스 드라이버(1130-1132)를 첨부할 수 있다. 그래서, 상기 입출력 관리자(1120)는 타겟 디바이스 드라이버(1130-1132)에 대해서 의도된 입출력 요청 패킷을 상기 타겟 디바이스 드라이버(1130-1132)에 첨부된 상기 파일시스템 감시자(1150)에 발송한다. 이 실시예에서, 상기 파일시스템 감시자(1150)는 다수의 파일시스템 드라이버 오브젝트(1130) 각각에 첨부한다.

도 13은 스택 포지셔닝을 이용하여 파일시스템 감시자(1250)에서 데이터보안을 제공하기 위한 방법의 일 실시예의 플로우차트이다. 실례의 목적으로서, 일 실시예는 상기 파일시스템 감시자(1250)이 파일시스템 요청을 제거하는 것으로 도시되어 있다. 상기 파일시스템 요청은 파일시스템에 명령되어지는 입출력 요청이다. 그러나, 본 발명이 파일시스템이나 감시하는 기능에 상관없는 그러한 디바이스 드라이버의 목적에 상관없이 어떠 디바이스 드라이브에서도 유용하게 실행될 수 있다.

도 13에 나타낸 바와 같이, 본 발명의 스택 포지셔닝 프로세스는 단계 1200에서 파일시스템 요청이 감지될 때마다 초기화된다. 단계 1210에서, 상기 프로세스는 파일시스템 필터가 단계 1223의 이전 반복에 의해 셧다운(shut down)되는지 않되는지를 결정한다. 그렇다면, 상기 프로세스는 단계 1225에서 계속되며 연속적인 호출이 필터되지 않는 것을 허가하고, 단계 1233에서 나간다.

단계 1210으로 돌아가서, 상기 파일시스템 필터가 이전에 셧다운되지 않는다면, 상기 프로세스는 단계 1220에서 계속되고 이것이 단계 1200에 의해 감지된 첫번째 파일시스템 요청인지를 결정한다. 그렇다면, 상기 프로세스는 단계 1230에서 계속되고 호출 모듈 어드레스(calling module address)를 결정한다. 이것은 첫번째 파일시스템 요청이기 때문에, 상기 파일시스템 감시자(1250)가 로드(load)된 것이 논리적으로 보증되고, 그러므로 상기 파일시스템과 관계된 디바이스 드라이버의 스택에서 기능적으로 최상위에 있다. 윈도우즈 NT 실시예에서, 상기 호출 모듈은 상기 파일시스템 감시자(1250)의 첨부된 디바이스부의 레퍼런스에 의해 결정되고, 상기 파일시스템 감시자(1250)이 디바이스 드라이버의 스택에서 기능적으로 최상위에 있을 때 제로(" zero" : 영)가 될 것이다. 단계 1231에 연속하여, 상기 프로세스는 단계 1230에서 결정된 호출 모듈 어드레스를 저장한다. 상기 프로세스는 단계 1232에서 계속되고 필터링 기능이나 파일시스템 감시자(1250)을 기반으로 하는 다른 유용한 기능을 완료한다. 상기 프로세스는 그때 단계 1233에서 끝낸다.

단계 1220으로 돌아가서, 이것이 단계 1200에서 감지된 상기 첫번째 파일시스템 요청이 아니라면, 상기 프로세스는 단계 1221에서 계속되고 호출 모듈 어드레스를 결정한다. 윈도우즈 NT 실시예에서, 상기 호출 모듈은 상기 파일시스템 감시자(1250)의 첨부된 디바이스부의 레퍼런스에 의해 결정되고, 어떤 디바이스 오브젝트가 상기 파일시스템 감시자(1250)에 첨부되었다면 영이 아닌 값(non-zero)이 될 것이다. 단계 1222에서, 상기 호출 모듈 어드레스는 단계 1

231에서 이전에 저장된 어드레스와 비교된다. 이러한 점에서 상기 호출 모듈 어드레스와 비교함으로서, 상기 파일시스템 감시자(1250)는 상기 호출 모듈과 상기 파일시스템 감시자(1250) 사이에서 삽입하므로서 상기 파일시스템 감시자(1250) 위에서 우선권을 얻고자 하는 어떤 다른 드라이버도 감지할 수 있을 것이다. 상기 두 호출 모듈 어드레스가 단계 1222에서 똑같다면, 상기 프로세스는 단계 1232에서 계속된다.

상기 두 호출 모듈 어드레스가 단계 1222에서 똑같지 않다면, 상기 프로세스는 단계 1223에서 계속되고, 필터링의 플래그 표시 셧다운(flag indicating shutdown)을 설정한다. 이 플래그는 단계 1210에서 검사된 똑같은 플래그이다. 상기 프로세스는 단계 1224에서 계속되고 리-훅(re-hook) 프로세스를 초기화하며, 도 14를 참조하여 아래에서 설명된다. 단계 1224 후에 상기 프로세스는 단계 1225에서 계속되며 연속적인 호출이 여과되지않는 것을 허가하고, 단계 1233에서 나간다.

도 14는 리-훅 프로세스의 일 실시예의 플로우차트이다. 단계 1300에서, 본 발명의 리-훅 프로세스는 초기화된다. 단계 1310에 연속하여, 상기 프로세스는 리-훅 처리가 얼마나 많이 시작되었는지가 저장된 카운트를 증가시킨다. 단계 1320에서 연속하여, 상기 프로세스는 상기 저장된 카운트가 최상 문턱에 도달했는지를 결정한다. 상기 최상 문턱은 프로그램할 수 있는 보안 응답을 트리거한 설결된 값이다.

데이터보안을 방해하는 시도로 제로(zero) 오차가 있는 실시예에서, 최상 문턱값만이 적당할 것이다. 몇몇 오차가 허용될 수 있는 실시예에서, 그 보다 큰 최상 문턱값이 절당할 것이다. 교대의 실시예에 있어, 다중 또는 교대의 프로그램할 수 있는 보안 응답은 다른 문턱에서 가능하게 될 것이다.

프로그램할 수 있는 보안 응답의 넓은 범위가 바람직하다. 프로그램 할 수 있는 보안 응답은 제한없이 요청된 데이터의 파괴 또는, 그런 데이터가 보호 가상파일시스템에 저장된다면 상기 데이터와 상기 보호 가상파일시스템 모두에 대한 파괴를 포함한다. 게다가 프로그램할 수 있는 보안 응답의 실시예는 열린 애플리케이션의 종결, 데이터 저장장치에서 파일시스템 감시자에 대한 파괴, 컴퓨터 작동에 대한 정지, 그리고 컴퓨터가 재부팅하게 하는 것을 포함한다. 전술된 것은 단지 실례이고, 본 발명에 따른 여러가지 프로그램할 수 있는 보안 응답을 제한하기 위한 뜻으로 파악되지는 않는다.

단계 1320으로 돌아가서, 상기 저장된 카운트가 최상 문턱에 도달했다면, 단계 1330에서 상기 프로그램할 수 있는 보안 응답은 초기화되고, 상기 프로세스는 단계 1350에서 종결된다.

상기 저장된 카운트가 최상 문턱에 도달하지 않았다면, 단계 1340에서 상기 프로세스는 상기 파일시스템 감시자(1050)의 재연결을 초기화한다. 이는 스택의 기능적으로 최상의 레벨에서 새로운 파일시스템 감시자(1050)를 초기화하므로서 이루어질 수 있다. 윈도우즈 NT와 같은 객체지향 환경에서, 싱글 디바이스 오브젝트에 많은 연결이 있을 수 있고, 상기 파일시스템 감시자(1050)를 상기 스택으로부터 삭제하거나 분리하는 것이 바람직할지라도 그렇게 할 필요는 없다. 교대의 실시예에서, 재연결은 기능적으로 최상의 레벨에서 이전에 초기화된 파일시스템 관리자(1050)을 리턴하도록 하는 상기 호출 체인(calling chain)을 편집하므로서 이루어질 수 있다.

단계 1340에서, 필터링이 다시 시작되도록 표시하기 위하여, 상기 프로세스는 또한 필터링의 플래그 표시 셧다운(flag indicating shutdown) 설정을 해제한다. 도 13에 도시된 바와 같이, 이 플래그는 단계 1123에서 설정되고 단계 1210에서 검사되는 똑같은 플래그이다. 상기 프로세스는 그때 단계 1350에서 종결된다.

또 다른 실시예에서, 본 명세서의 실시예에 따른 스택 포지셔닝의 형태는 도 6-8에 도시된 실시예와 같이 백-채널링 프로세스로 연결될 수 있다. 상기 최상의 스택 포지션은 상기 백-채널링이 호출 처리에서 직접적으로 실시되거나 중간 파일시스템 필터에서 직접적으로 실시되지 않는 것을 보증하는 것이 잇점이다.

본 발명에 따른 한층 더한 보호 메카니즘은 시계 작동을 감시하는 것이며, 아래에서 기술된다.

몇몇 컴퓨터 프로그램은 프로그램 사용시간을 한정하는 제한을 포함한다. 이러한 제한은 라이센스로 규정되거나, 프로그램의 사용상태에 대해 동의되거나, 프로그램의 소유자나 제공자에 의해 규정지어질 수 있다. 예를 들어, 일반적으로 소프트웨어의 트라이얼 버전(trial version)은 예를 들어 30일이라는 정해진 기간동안만 사용을 허가한다. 트라이얼 소프트웨어의 만료기간을 설정하는데 대한 알려진 방법중 하나는 시스템의 시계정보를 기록하는 것이다. 상기 시계정보는 날짜와 시계정보를 포함하며, 설치한 날로부터 30일이 되는 만료날짜를 설정한다. 또 다른 방법으로는 컴퓨터 코드에서 프로그램된 만료날짜를 가지는 것이다. 이러한 방법상에서, 프로그램은 시스템 시계 날짜가 프로그램이 컴퓨터 시스템에 설치될 때와 상관없이 만료날짜보다 더 빠르다면 작동할 것이다.

이러한 두 가지 방법에 따라, 사용자가 만료날짜 이후에 컴퓨터 프로그램을 실행시키는 것을 방지한다. 일반적으로, 만료날짜 이후에 프로그램을 실행시키려는 시도는 사용자에게 프로그램이 만기가 되었다는 스크린-메세지로 알려준다. 만료날짜 매개변수를 회피하기 위한 알려진 방법은 시스템 시계의 날짜와 시간을 수동으로 재설정하는 것이다. 즉, 시간과 날짜를 만료날짜모다 빠르게 설정하는 것이다.

시도는 시스템 시계 날짜가 만료기간 이후가 되었을 때 사용자가 프로그램 실행을 시도한 후에 시스템의 시계정보에 독립적으로 프로그램을 완전히 사용할 수 없도록 하므로서 이러한 문제를 처리해왔다. 그러나, 이러한 해법은 사용자가 시스템 시계를 정정하고 소프트웨어의 허가된 사용을 계속하는 것을 허락하지 않는다는 점에서 가혹하다.

컴퓨터 시스템 분야에서, 영원히 프로그램이나 날짜를 사용하거나 엑세스할 수 없도록 하지 않으면서, 만료 매개변수와 같은 컴퓨터 코드로 임베드된(embedded) 사용자-제한의 속임을 방지하기 위한 필요성이 있다. 본 발명은 날짜에 대한 권한없는 접근을 방지하기 위하여 시스템 시계를 감시하는 방법과 장치를 제공하므로서 이러한 필요성을 처리한다. 본 발명은 시스템 시계에 대한 변경 또는 개조를 감지하고 시스템 시계가 그것의 정확한 값으로 리턴될 때까지 시계에 관한 승인에 대한 기능을 억제한다.

도 15는 본 발명의 바람직한 실시예에 따른 컴퓨터의 시스템 시계를 감시하기위한 방법을 나타내는 블록다이어그램이다. 이러한 방법에 따라서, 시계 감시자는 메모리상에 저장된 승인 데이터베이스와 통신한다. 상기 승인 데이터베이스는 적어도 하나의 시계-관계의 승인부와 저장된 시간값을 가지는 저장된 시간값 필드를 포함한다. 상기 시계-관계의 승인은 시스템 시계로부터 정보(시간과 날짜정보 또는 시스템 시계의 속도)에 기초한 날짜에 대한 접근을 제어하는 매개변수이다. 컴퓨터 시스템은 대체적으로 시스템 시계에 따라 작동한다는 것은 본 기술분야에 공지되어 있다.

예를 들어, 시계-관련의 승인은 사용자가 얼마나 오래동안 날짜, 시간 또는 분에 의한 데이터에 접근할 수 있는지를 명기한 것을 포함한다. 이러한 승인을 실행시키는 한가지 방법은 상기 프로세스가 만료되는 시간까지 처리가 상기 데이터에 접근하는 시간으로부터 접근시간을 증가시키는 것이다. 상기 접근시간은 내부적으로 감지된다; 그러나, 시스템 프로세서의 속임수로부터 유도된 시스템 시계의 속도는 접근시간을 감소시킨다. 상기 프로세스 동안, 접근시간이 완전히 끝나면, 상기 프로세스는 자동적으로 종결된다. 접근시간이 만료된 후, 상기 데이터는 덮어쓰기되고 삭제된다. 시계-관련의 승인의 다른 예는 상기 데이터가 더 이상 이용할 수 없게 되는 날짜를 명기하는 것이다. 사용자는 만료날짜까지 상기 데이터에 접근한다. 프로세스가 만료날짜에서 상기 데이터를 열게된다면, 상기 프로세스는 자동적으로 종결되고 상기 데이터는 덮어쓰기되고 삭제된다. 시계-관련의 승인의 다른 예가 상기 데이터에 접근될 수 있는 권한있는 날짜를 명기하고 있다. 이러한 승인에 따라, 사용자는 권한있는 날짜가 통과될 때까지 상기 데이터에 접근할 수 없다. 일단 권한있는 날짜가 통과되면, 사용자는 상기 데이터에 접근하게 될 것이다. 시계-관련의 승인의 다른 예는 " 승인없음" 또는 " 제한없는 사용" 을 포함한다. 상기 시계-관련의 승인에 대한 데이터는 하드디스크, 플로피디스크, CD-ROM, 자기테이프와 같은 저장장치에 저장되거나, 승인 데이터베이스나 컴퓨터 프로그램에 임베디드된다.

상기 시계 감시자는 실행시 단계 1412에서 시계 감시자를 초기화시키는 컴퓨터 코드로 구성된다. 본 발명의 일 실시예에서, 컴퓨터 코드는 읽을 수 있는 매체(예를 들어 자성디스크 또는 광학디스크)에 저장된 컴퓨터 프로그램이다. 컴퓨터 코드는 코드의 싱글 모듈(single module)이 될 수 있고, 바람직하게는 시계 감시자의 기능을 수행하기 위한 연속의 모듈로 분해될 수 있다. 일 실시예로서, 일반적인 기능은 코드의 모듈의 기능을 수행하기위하여 호출 명령을 내리므로서 코드의 싱글 모듈에 의해 한 번 이상 실행될 수 있다. 초기화상태에서, 상기 시계 감시자는 시스템 시계로부터 현재 시간을 읽는다(단계 1414). 상기 시계 감시자는 그때 이전의 경우에서 초기화 되었는지를 포함하여, 승인 데이터베이스가 초기화되는지를 결정한다(단계 1416). 본 발명의 일 실시예에서, 컴퓨터 프로그램의 실행은 승인 데이터베이스가 승인 데이터베이스 드라이버에 의하여 초기화되는 것의 원인이 된다. 상기 승인 데이터베이스 드라이버가 컴퓨터 시스템에 로드된다면, 상기 데이터베이스는 " 초기화" 를 고려하게 된다. 이러한 실시예에서, 상기 시계 감시자 컴퓨터 코드는 상기 승인 데이터베이스를 초기화시키기위한 컴퓨터 프로그램내에 임베디드될 수 있다. 다른 실시예에서, 상기 승인 데이터베이스가 초기화되는지를 결정하는 단계는 상기 승인 데이터베이스로부터 저장된 시간값의 읽기를 포함한다. 상기 저장된 시간값이 제로(zero: 0)이면, 그때 상기 데이터베이스는 초기화되지 않는다. 상기 저장된 시간값이 제로가 아니라면, 그때 상기 데이터베이스는 초기화된다.

상기 승인 데이터베이스가 초기화되면, 그때 단계 1414로부터 현재 시간값은 상기 승인 데이터베이스의 상기 저장된 시간값 필드에서 저장된 시간값과 비교된다(단계 1418). 단계 1414로부터 현재 시간값이 저장된 시간값보다 늦은 시간이라면, 상기 현재 시간값은 상기 저장된 시간값 필드에 저장된다(단계 1420). 단계 1414로부터 현재 시간값이 저장된 시간값보다 더 이르다면, 상기 시간-관련의 승인 기능이 억제되고, 그 때문에 상기 데이터에 대한 접근이 차단된다(단계 1422).

상기 승인 데이터베이스가 초기화되지 않는다면, 단계 1414로부터 시스템 시계의 현재 시간값은 저장된 시간값 필드에 저장된다(단계 1224). 단계 1420, 1422 또는 1424 이후에, 도 15에 도시된 방법에 따라, 상기 시계 감시자는 초기화된다(단계 1426).

도 16은 본 발명의 일 실시예에 따른 방법을 나타낸다. 상기 시계 감시자가 초기화된 이후에, 상기 시스템 시계는 상기 시스템 시계의 무결성을 증명하기위하여 선결된 트랙킹 간격(tracking interval)에서 검사된다. 상기 간격 검사는 단계 1528에서 초기화된다. 간격 검사는 시스템 시계의 속도를 이용하여 단계 1414로부터 현재 시간값을 증가시키고 정확한 시스템 시간을 찾아낸다. 정확한 시스템 시간은 승인 데이터베이스에서 저장된 시간값과 시계 감시자가 초기화되는 시간으로부터 측정된 내부의 경과한 시간을 합한 것이다. 상기 내부의 경과한 시간은 바람직하게 상기 승인 데이터베이스에 저장된다. 선결된 트랙킹 간격에서, 상기 시계 감시자는 시스템 시계의 시간값을 읽고(단계 1530), 내부 시계에 의해 유지되는 정확한 시스템 시계에서 그것과 비교한다(단계 1532). 상기 선결된 트랙킹 간격은 어떤 시간값도 될 수 있으나, 바람직하게는 0~60초이다. 더 바람직하게, 상기 선결된 트랙킹 간격은 1분이다. 단계 1530에서 읽은 바와 같이, 시스템 시계의 시간값과 상기 정확한 시스템 시간 사이의 비교를 기초로, 상기 시계 감시자는 또한 단계 1532에서 시간 이탈을 일으킨다. 상기 시간 이탈은 용인되는 이탈과 비교된다. 상기 용인되는 이탈은 바람직하게 선결되고 시간의 어떤 값(예를 들어, 0초~3시간 범위내의 시간값)도 될 수 있다. 가장 바람직하게, 용인되는 이탈은 3시간이다.

시간 이탈이 용인되는 이탈 밖에 있다면, 상기 시계 감시자는 상기 시스템 시계값이 변경되고 모든 시계-관련의 승인 기능을 억제하도록 결정된다(단계 1534). 상기 시간 이탈이 용인되는 이탈내에 있다면, 상기 시계 감시자는 상기 시계-관련의 승인을 실행한다(단계 1536). 단계 1538에서, 상기 정확한 시스템 시간이 바람직하게 상기 승인 데이터베이스에 저장되고, 상기 간격 검사는 완료된다(단계 1540).

일 실시예에서, 도 16의 단계는 상기 시계-관련의 승인이 단계 1534 또는 도 15의 단계 1422에서 기능이 억제되면 일반적으로 반복된다. 도 16에서 단계의 이러한 반복으로 발생되는 시간 이탈이 용인되는 이탈내에 있으면, 상기 시계-관련의 승인은 재허가되고, 승인은 강요되며, 필요에 따라 상기 정확한 시스템 시간은 바람직하게 상기 승인 데이터베이스의 저장된 시간값 필드에 저장된다. 그리고 상기 시스템 시계로부터 읽어진 시간값은 상기 승인 데이터베이스의 최근에 알려진 좋은 시스템 시간 필드에 저장된다.

다른 실시예에서, 컴퓨터가 셧다운(shut down) 되었을 때, 상기 정확한 시스템 시간은 시스템 시계값과 비교된다. 상기 시스템 시계값이 인정된 이탈내에 있으면, 상기 시스템 시계값은 상기 저장된 시간값 필드에 저장된다. 이는 상대적으로 설명되고 저장되는 작은 이탈을 가능하게 한다.

다른 보호 메카니즘은 포트 블록킹(port blocking)이다. 컴퓨터 시스템에서 프로세스는 시리얼 포트나 인터넷 연결과 같은 많은 시스템 리소스를 엑세스할 수 있다. 보호 데이터가 비보호 애플리케이션에 의해 엑세스되고 있는 상황에서, 이러한 방법은 비보호 애플리케이션이 데이터보안을 손상시키는 작동 수행으로부터 제한될 수 있도록 개발되어야 한다.

그것은 통신의 외부로부터 완전히 격리된 시스템에서 보호데이터를 여는 것으로 알려져 있다. 즉, 상기 시스템은 사고 또는 파괴에 의해서 비보호 애플리케이션이 비보호 데이터를 손상시키는 수단에 연결되어 있지 않다. 그것은 또한 보호 애플리케이션으로 보호 데이터를 오픈하는 것으로 알려져 있으며, 보호 데이터를 손상시키는 사고나 파괴의 위험으로부터 자유롭게되는 것이다. 이러한 해법은 일반의 소프트웨어 애플리케이션의 사용이 보호 데이터를 여는 것이나, 외부 통신으로부터 연결되지 않은 컴퓨터의 사용을 차단하고, 그러므로 그들의 유용성에 제한을 한다.

상기 기술된 실시예는 작동중인 컴퓨터의 포트 리소스를 이용하는 것으로부터 어떤 프로세스를 불가능하게 한다. 이는 보호데이터를 여는 거와 같은 프로세스를 안전하게 할 수 있다. 본 발명의 바람직한 실시예로, 보호된 프로세스의 상태는 보호 프로세스의 리스트상의 프로세스가 존재하므로서 결정된다.

도 17에 도시된 바와 같이, 바람직한 실시예로 컴퓨터(1600)에서 제어 애플리케이션(1610)은 커널(링 0) 레벨(1620)에서 작동하고 애플리케이션(1630)은 더 높은 레벨(1640)에서 작동한다. 애플리케이션이 포트(1650)에 대한 접근을 요청할 때, 제어 애플리케이션(1610)은 이러한 액세스 요청을 감시하고 처리한다.

도 18에 도시된 바와 같이, 어떤 컴퓨터 시스템(예를 들어, 마이크로소프트 윈도우즈 NT와 윈도우즈 2000 운영체제)에서, 포트 감시는 모든 포트-관련의 호출을 차단할 수 있다. 포트 요청이 초기화될 때(1700), 제어 애플리케이션(1610)은 그 요청을 차단하고, 프로세스 아이디를 결정한다(1710). 바람직한 실시예로서 상기 제어 애플리케이션(1610)은 포트 여는 것이 허락되지 않은 프로세스 리스트를 엑세스한다. 상기 프로세스 아이디는 상기 프로세스가 (포트 여는 것이 허락되지 않도록) 보호되는지를 결정하기 위해 이용된다(1720). 그것이 보호되지 않는다면, 그때 상기 요청은 포트에서 통과된다(1750).

도 19(a)에 도신된 바와 같이 어떤 컴퓨터 시스템(예를 들어, 마이크로소프트 윈도우즈 95와 98 운영체제)에서, 상기 포트 감시는 열기와 닫기 호출만을 차단할 수 있다. 포트를 엑세스하는 프로세스가 그때 보호 프로세스가 되지 않는다는 것을 보증하기 위하여, 검사는 보호되는 어떤 프로세스에서도 실행되어야만 한다. 열기 포트 요청이 초기화되었을 때(1810), 제어 애플리케이션(1610)은 그 요청을 차단하고, 프로세스 아이디를 결정한다(1810). 바람직한 실시예에서 상기 제어 애플리세이션(1610)은 포트 열기를 허가하지 않는 프로세스 리스트를 엑세스한다. 상기 프로세스 아이디는 상기 프로세스가 (포트 열기를 허락하지 않도록) 보호되는지를 결정하기위해 사용된다(1820). 그것이 보호된다면, 상기 요청은 봉쇄된다(1830). 그것이 보호되지 않는다면, 그때 상기 요청은 상기 포트에서 통과되고 상기 프로세스 아이디와 포트 프로세스는 탐지된다(1850).

도 19(b)에 도시된 바와 같이, 닫기 포트요청은 초기화될 때(1860), 제어 애플리케이션(1610)은 그 요청을 차단하고, 호출을 완료한다(1862). 그때 상기 프로세스 아이디와 포트 처리는 탐지된 열린 포트의 데이터베이스로부터 제거된다(1864).

도 19(c)에 도시된 바와 같이, 열기 포트와 닫기 포트 요청에서 이러한 작동에 추가적으로, 프로세스가 그것이 보호될 것인지를 결정하는 보안검사를 받게될 때(1870), 그것의 프로세스 아이디는 탐지된 열기 포트의 데이터베이스에 대해서 검사된다(1872). 상기 프로세스가 열기 포트를 가지고 있다면, 상기 프로세스는 안전할 수 없게 되고 상기 보안검사는 실패하고(1874), 상기 보안검사는 완료된다(1878). 상기 프로세스가 열기 포트를 가지고 있지 않다면, 그것은 보안검사를 통과하고 프로세스 아이디는 보호 프로세스 리스트에 추가될 것이다(1878).

본 발명의 더 나은 실시예는 본 명세서에서 제공된 방법에 따라 어떤 프로세스가 포트를 사용하는 것을 제한받는다는 점에서 포트 블록킹 시스템(port blocking system)로 향하게 된다. 앞으로 기술된 것은 본 명세서에서 제공된 방법에 따라 어떤 프로세스가 포트를 사용하는 것으로부터 제한하는 포트 블록킹 구성요소를 가지는 보호데이터 전송시스템이다. 또한 앞으로 기술된 것은 본 명세서에서 제공된방법에 따른 포트 이용을 차단하기 위해 프로그램된 컴퓨터의 읽을 수 있는 매체이다. 또한 앞으로 기술된 것은 어떤 프로세스가 본 명세서에서 제공된 방법에 따른 포트를 이용하는 것으로부터 차단하는 포트 블록킹 시스템을 포함하기 위하여 형성된 컴퓨터이다.

포트 블록킹과 유사한 또 다른 보호 메카니즘은 네트워트/TDI 블록킹이다.

실시예는 어떤 프로세스가 작동하고 있는 컴퓨터의 네트워크 리소스를 이용하는 것으로부터 차단한다. 이것은 프로세스(예를 들어 보호데이터를 여는 것)를 안전하게 한다. 본 발명의 바람직한 실시예로, 보호되는 프로세스의 상태는 보호 프로세스 리스트에서 프로세스 존재에 의해서 결정된다.

도 20에 도시된 바와 같이, 바람직한 실시예로 컴퓨터(1900)에서, 제어 애플리케이션(1910)은 커널(링 0) 레벨(1920)에서 작동하고 애플리케이션(1930)은 상위레벨(1940)에서 작동한다. 애플리케이션이 네트워크/TDI 인터페이스(1950)을 엑세스할 때, 제어 애플리케이션(1910)은 이러한 액세스 요청을 감시하고 처리한다.

도 21에 도시된 바와 같이, 네트워크 블록킹은 보호 애플리케이션에 대하여 처리되도록 하기 위한 보내기 요청을 허가하지 않으므로서 이루어진다. 보내기 요청이 초기화될 때(2000), 제어 애플리케이션(1910)은 그 요청을 차단하고, 프로세스 아이디를 결정한다(2010). 바람직한 실시예로 상기 제어 애플리케이션(1910)은 네트워크 엑세스를 허락하지 않는 프로세스 리스트를 엑세스한다. 상기 프로세스 아이디는 상기 프로세스가 (네트워크 엑세스를 허락하지 않도록) 보호되는지를 결정하기 위하여 사용된다. 그것이 보호된다면, 상기 요청은 단계 2030에서 차단된다. 그것이 보호되지 않는다면, 그때 상기 요청은 상기 네트워크를 통과하게 된다(2050).

본 발명의 더 나은 실시예는 본 명세서에서 제공된 방법에 따라 어떤 프로세스가 네트워크를 엑세스하는 것으로부터 제한된다는 점에서 네트워크 블록킹 시스템으로 향하게 된다. 앞으로 기술된 것은 본 명세서에서 제공된 방법에 따라 어떤 프로세스가 네트워크를 엑세스하는 것으로부터 차단하는 네트워크 블록킹 구성요소를 가지는 보호데이터 전송시스템이다. 또한 앞으로 기술된 것은 본 명세서에서 제공된 방법에 따라 네트워크 이용을 차단하기 위해 프로그램된 컴퓨터에서 읽을 수 있는 매체이다. 또한 앞으로 기술된 것은 본 명세서에서 제공된 방법에 따라 어떤 프로세스가 네트워크를 엑세스하는 것으로부터 차단시키기 위한 네트워크 블록킹 시스템을 포함하기 위해 구성된 컴퓨터이다.

또 다른 보호 메카니즘은 파일 숨기기나 파일 은폐시키기와 관계된다. 도 22는 시스템 호출을 감시하므로서 파일을 숨기거나 파일을 은폐시키기 위한 시스템의 일 시시예의 블록다이어그램이다. 도시된 바와 같이, 상기 시스템은 운영체제(OS)층(2210)과 애플리케이션층(2112)을 포함한다. 일 실시예에서, 포인트가 파일시스템 제어로 들어감을 허락하는 윈도우즈 98, NT, 2000, 유닉스기반 시스템 또는 다른 운영체제가 또한 본 발명의 여러가지 실시예의 똑같은 원리를 이용할 수 있지만, 운영체제층(2100)은 윈도우즈 95이다. 운영체제(2100)는 애플리케이션층(2112)에서 애플리케이션으로부터 호출을 처리하는 파일시스템(2114)(설치가능한 파일시스템, IFS)을 포함한다. 일반적으로 디바이스 드라이버는 운영체제(2110)에서의 디바이스가 되고, 애플리케이션층(2112)에서의 애플리케이션은 정의된 애플리케이션 프로그램 인터페이스(API)를 통한 작업을 수행하기 위한 디바이스를 이용한다. 본 발명의 원리에 따라서, 고 레벨 은폐(cloaker) 애플리케이션(2116)은 은폐 드라이버(2118)와 관련된다. 예를 들어, 은폐 애플리케이션(2116)과 은폐 드라이버(2118)는 어떤 소프트웨어 벤더(vendor)에 의해 전해진 구성요소가 될 수 있고 아래에서 기술된 바와 같은 파일 숨기기나 파일 은폐시키기와 같은 기능을 수행하기 위해 협력될 수 있다.

은폐 드라이버(cloaker application)(2118)는 파일시스템(2114)을 통해 통과하는 호출을 감시하고 반응하도록 작동한다. 은폐 애플리케이션(2116)에 추가로, 다른 고 레벨 애플리케이션(2119)는 애플리케이션층(2112)에 있을 수 있고 파일시스템(2114)을 통해 호출을 보낼 수 있다. 예를 들어, 그러한 호출은 워드프로세싱 또는 다른 애플리케이션으로부터 또는 윈도우즈 파일 매니저 환경내로부터 시작할 수 있다. 여기에서 중요한 파일시스템 호출은 " 열기" , " 처음 찾기" , " 다음 찾기" , " 삭제" , 그리고 " 이름바꾸기" 의 명령과 관련된다. 간단한 설명으로, 이러한 명령은 또한 다소 간단하게 호출로서 언급된다. 일반적으로, 중요한 작동은 파일을 열기, 찾기 또는 삭제하기의 기본적인 어떤 작동을 포함한다. 그러므로, 예시된 방법으로, " 읽기" 와, " 쓰기" , " 속성 얻기" 호출은 " 열기" 호출 내에 본 발명의 실시예 대한 목적으로 포함된다. 그 후, 상기 " 열기" 호출은 각각의 이러한 호출로 발생될 것이다. 유사한 방법으로, " 디렉토리" 호출은 " 처음 찾기" 와 " 다음 찾기" 호출내에 포함된다. 각각의 미디어 디바이스(2120)(하드디스크 드라이브, ZIP 드라이브, 플로피드라이브, 테이프 드라이브, 쓰기가능한 CD-ROM 드라이브, 고정된 또는 이동가능한 저장장치)는 일반적으로 미디어 디바이스(2120)와의 인터페이스를 처리하는 저 레벨 드라이버(2122)와 관계된다.

작동상에서, 은폐 또는 감시 드라이버(2118)는 파일시스템(2114)를 통과하는 각각의 호출을 얻고 분석한다. 특별한 디바이스와 실제로 관련되지 않는 디바이스 드라이버로 파일시스템(2114)에 접속하므로서 은폐 드라이버(2118)는 파일시스템 호출에 대한 엑세스를 얻을 수 있다. 이러한 엑세스를 이용하여, 은폐 드라이버(2118)는 상기 호출이 은폐 애플리케이션(2116)이나 다른 애플리케이션(2119)로부터 일어났는지 그리고 상기 호출이 " 열기" , " 처음 찾기" , " 다음 찾기" , " 삭제" , " 이름바꾸기" 와 같은 중요한 호출인지를 엑세스할 수 있다. 은폐 드라이버(2118)는 또한 미디어 디바이스(2120)과 관련된 드라이버(2122)에서 호출을 통과시킬지 또는, 상기 호출을 거절 또는 중지시킬지 또는, 다음의 층 아래로 상기 호출을 통과시키지 않고 리턴 변수를 바꾸거나 리턴 변수를 발생시킬지를 결정할 수 있다. 그러한 작동으로, 은폐 드라이버(2118)은 사용자가 완전히 볼 수 없도록 하는 그러한 방법으로 사용자로부터 파일을 숨길 수 있고 어떠한 애플리케이션 또는 윈도우즈 시스템 명령으로부터 찾을 수 없을 것이다.

은폐 애플리케이션(2116)은 사용자가 파일을 은폐키시거나 파일을 숨기는 것을 허가하도록 실행된다. 은폐 애플리케이션 작동은 사용자가 은폐를 위해 요구되는 파일을 선택하는 것을 허가하고 그때 은폐 드라이버(2118)에서 선택된 파일이름을 통과시키므로서 이루어진 그러한 파일에서 은폐 작동을 적용할 수 있도록 허가한다. 그래서, 그것들은 은폐 드라이버(2118)내의 룩업(lookup) 테이블 또는 데이터베이스에 추가된다. 바람직하게, 사용자는 은폐 프로세스를 초기화하기 전에 암호를 선택할 것을 요청받고 이 암호만이 사용자가 이전의 은폐 파일을 은폐에서 풀도록 하는 것을 허가한다. 가장 바람직하게, 상기 암호는 윈도우즈 사용자 암호와는 다르다. 이러한 방법으로 둘 이상의 사용자는 같은 컴퓨터 하드웨어를 사용하지만 저장된 파일의 다른 부분을 엑세스한다. 사용자1은 사용자2로부터 시스템상의 기밀의 숨은파일을 보존한다. 숨은 파일의 내용뿐만아니라, 파일의 존재도 숨는다. 본 발명의 일 실시예에서, 은폐 파일은 디렉토리 리스트에서 보여지지 않을 뿐만아니라 적당한 암호를 입력해야만 하는 사용자에 의해 은폐가 해제될 때까지 운영체제에 의해서 어떤 방법에서도 엑세스될 수 없다.

추가된 안전한 예방조치로서, 은폐된 파일은 파일이름이 은폐 드라이버의 룩업이나 데이터베이스에 추가된 후(또는 전)에 자동적으로 암호화될 수 있다. 정확한 암호의 입력후, 암호화된 파일은 파일 열기에서 복호화되고 파일 닫기에서 다시 암호화된다. 은폐 파일(cloaked file)은 더 이상 은폐될 필요가 없을 때까지 닫기될 때 암호화된다. 권한있는 사용자가 은폐 애플리케이션(2116) 이용에서 은폐를 제거할 때, 은폐 드라이버 룩업 테이블 또는 데이터베이스로부터 파일이름을 제거하므로서 실행되며, 파일은 자동적으로 복호화될 것이다. 은폐 애플리케이션과 은폐 드라이버가 암호화와 복호화 프로세스에 영향을 미칠 수 있는 많은 방법이 있고 상기된 설명은 단지 일 실시예이다. 암호화의 기본적인 유용성은 은폐 드라이버 프로그램이 실행되지 않는 다른 운영체제에서 시스템을 작동시키기 위해서 플로피디스크로부터 시스템을 부팅시키므로서 기인된다. 그래서, 파일 암호화는 비록 플로피 부트 상황에서 복호화가 불가능하지 않더라도 그 풀기 힘든 내용을 번역하므로서 파일에 대해 권한없는 접근을 막는다.

도 23은 운영체제의 파일시스템 구조에서 파일시스템 호출을 감시하기 위한 그리고 최초의 호출 또는 리턴된 변수를 제어하기 위한 본 발명에 따른 방법의 실시예의 플로우차트이다. 도 23의 방법은 미디어 드라이버(2220)에 저장될 수 있는 파일 숨기기 또는 파일 은폐시키기를 목적으로 가상 은폐 소프트웨어 드라이버(2118)에 의해 실행된다. 일 실시예로, 도 23의 방법은 윈도우즈 95의 설치가능한 파일시스템(IFS)내에서 실행되는 벤더 공급 드라이버(vender supplied dirver : VSD)를 이용하여 실행될 수 있다.

도 23에 도시된 바와 같이, 단계 2210에서, 파일시스템 호출은 차단된다. 일반적으로 상기 파일시스템 호출은 미디어 디바이스(2220)에 저장된 데이터의 측면에서 어떤 기능을 수행하는 경향이 있으나 그 기능을 완료시킬 수 있기 전에 차단된다. 도 23의 단계 2212에서, 차단된 호출의 타입이 처리되는 것인지가 결정된다. 일반적으로, 중요한 호출은 사용자 또는 이전 사용자가 은폐시킨 파일을 찾거나 확인할 수 있는 어떤 호출을 구성한다. 은폐시키므로서, 파일은 사용자 관점과 운영체제의 관점으로부터 볼 수 없도록 변형되며 엑세스될 수 없고, 교환될 수 없고, 변형될 수 없고, 삭제될 수 없고, 목록으로 나타날 수 없고, 운영체제에 의해서 조차 찾을 수 없다. 그러므로, 일단 파일이 은폐되면, 중요한 호출은 " 열기" , " 처음 찾기" , " 다음 찾기" , " 삭제" , " 이름바꾸기" 를 포함한다. 일반적으로, 파일이 은폐된다면, 은폐된 파일이나 그것의 리턴된 변수를 참고로 어떤 동작을 수행하기위해 시도되는 호출은 파일을 숨기거나 은폐시기기 위하여 제어될 수 있다. 그러한 제어는 에러 코드나 도 26A-26G와 관련하여 아래에 설명된 다른 작동과 같은 리턴 변수 발생을 포함한다. 그러한 호출이 단계 2212에서 확인되지 않는다면, 그때 단계 2214에서 최초의 호출은 파일시스템을 통과하게 된다. 상기 호출이 처리되고 그것이 중요한 호출중 하나라면, 그때 단계 2216에서 호출 프로세싱의 어떤 타입은 은폐된 파일이 사용자에게 알려지지 못하도록 보증하기 위하여 실행된다. 프로세싱의 특별한 타입은 호출의 타입에 의존하고 도 26A-26G와 관련하여 더 충분히 설명된다.

본 발명에 따라서, 단계 2216의 호출 프로세싱은 사용자와 호출시스템 애플리케이션 모두에 있어 명확하게 이루어질 수 있다. 상기 호출 프로세싱 이후에, 상기 호출은 호출시스템 애플리케이션(예를 들어, 도 22의 애플리케이션(2219))으로 리턴된다(단계 2218).

도 24는 윈도우즈 95의 설치가능한 파일시스템(IFS)(2114)의 파일시스템 논리층의 다이어그램이다. 도 3에 도시된 바와 같이, 설치가능한 파일시스템은 완전해진다.;6f 블록-디바이스요청이 통과되는 하나 이상의 가상 디바이스를 각각 포함하는 32 논리층. 예를 들어, 하드디스크 드라이버에 대해, 파일시스템 요청(또는 호출)은 일반적으로 하드웨어상에서 약 다섯 개의 가상 디바이스를 통과할 것이다.

도 25는 윈도우즈 95 운영체제의 파일시스템 논리층내의 전형적인 파일시스템 요청 체인 또는 패스(path)의 다이어그램이다. 도 25에 도시된 바와 같이, 일반적인 패스(path)는 설치가능 파일시스템 관리자(2422)에서 시작하고 파일시

스템 드라이버(2424)로 이동한다. 그때 상기 요청은 특정 타입 드라이버(2426)과, 이러한 경우에, 벤더(vendor) 공급 은폐 드라이버(2118)로 이동한다. 벤더 공급 은폐 드라이버(2118) 후에, 상기 요청은 포트 드라이버(2122)와 미디어 드라이브(2120)(예를 들면, 하드드라이버 또는 다른 저장장치)로 떨어진다. 상기 요청이 물리층(physical level)에서 완료된 후, 상기 요청은 상기 체인을 상기 호출 시스템 애플리케이션으로 리턴한다.

도 24에서, 좌측의 숫자는 추상(abstraction)의 상위층을 나타내는 가장 작은 숫를 가진 추상층을 나타낸다. 최상층은 파일시스템에서 입구점(entry point)이다. 높은 수는 하드웨어에 더 밀접해지고 최상의 수(바닥층: bottom layer)는 하드웨어를 직접적으로 엑세스할 수 있는 가상드라이브를 나타낸다. 입출력(I/O) 수퍼바이저(supervisor)(IOS)은 파일시스템 계층을 통과하도록 하는 요청을 처리한다. 상기 체인상에서 각각의 가상디바이스는 상기 요청이 나아가게 되는 논리층 또는 물리층을 기반으로 하는 요청을 선택할 수 있다. 상기 디바이스는 또한 그것이 상기 애플리케이션에서 상기 체인 뒤로 통과하는 것처럼 요청의 결과를 볼 수 있다. 게다가, 상기 체인에서 상기 가상디바이스 드라이버(virtual device drivers: VxDs)는 스스로 요청을 서비스할 수 있고 그것을 하위 레벨로는 통과시킬 수 없다. 또는 그것들 자체로 요청을 발생시킬 수 있다.

프로세스는 설치가능한 파일시스템의 각 레벨에서 일어날 수 있으나, 대부분의 블록 디바이스(block devices)는 상기 체인의 각 레벨에서 엔트리(entry)를 요구하지 않는다. 논리층 구조의 최상층에서, 상기 설치가능 파일시스템 관리자층은 애플리케이션으로부터 고-레벨 입출력 요청을 처리한다. 그것은 특정 논리 드라이브에서 직접적인 호출을 하고 정확한 콜-다운 체인(call-down chain)을 적당한 볼륨 트랙커(volume tracker), 파일시스템 드라이버(file system driver: FSD), 등등으로 통과시킨다. 볼륨 트랙커는 동일한 이동가능 규칙을 가진 디바이스 그룹으로 작동한다. 예를 들어, CD-ROM 볼륨 트랙커는 그것이 어떤 요청이 하위 층을 통과하는 것을 허가하기 전에 파일시스템의 CD가 드라이브상에 있는 것을 보증한다. 파일시스템 드라이버(FSDs)는 하드디스크나 CD-ROM 디바이스와 같은 특정 타입의 모든 디바이스로 작동한다. 그것들은 설치가능 파일시스템 관리자에 의해 발생된 논리적 요청을 수신하고 하위레벨로 통과시키기 위해 그것을 물리적 요청으로 변환시킨다. 추가적으로, 파일시스템 디바이스는 디스크와 같은 디바이스에 대한 논리적 에러복구를 초기화할 수 있다.

특정 타입 드라이버(Type specific drivers: TSDs)는 특정 타입의 모든 디바이스로 작동한다. 그것들은 파일시스템 드라이버로 발생된 논리적 요청을 수신하고 그것을 물리적 부분 요청으로 변환시킨다. 그것들은 일반적으로 그것들의 통신관계의 파일시스템 드라이버로서 같은 층에 존재하지만 상기 체인보다 하위에 있다. 스카시-아이저(SCSI-izers)는 상기 체인의 다음에 있고 사용된다. 왜냐하면 스카시 드바이스는 일반적인 IDE/ESDI 디바이스와 같은 다른 디바이스보다 더 복잡한 요청 패킷을 요청하기 때문이다. 스카시-아이저(SCSI-izers)는 일반적인 물리적 요청을 수신하고 상세하게 포함하는 스카시 요청 블록(SCSI Request Block: SRB)과, 논리적 단위수(Logical Unit Number: LUN)와 타겟(스카시 타겟은 7 논리적 단위수를 호출할 수 있다)과 같은 상기 요청에 대한 스카시-특정 정보를 생성한다.

벤더 공급 드라이버는 윈도우즈 95상의 써드-부(third-party) 개발자를 위해 특정 층을 제공한다. 결론적으로, 벤더 공급 드라이버 층은 기능적으로 종래의 사용이 포함하는 벤더 공급 드라이버 작성자(writer)에 의해 결정된다.; 블록-디바이스 감시자, 저-레벨 제2의 디스크 캐쉬(예를 들어, 플래쉬 메모리에서의 캐쉬), 데이터 압호화, 레이드 디스크 관리(RAID disk management).

스카시 포트 드라이버는 요청을 수신하고 스카시 미니포트 드라이버가 그것들을 배치시키는 것을 결정한다. 다수의 스카시 타입은 주문형 스카시 미니포트 드라이버의 요구에 따라 같은 시스템에 로드될 수 있다. 상기 스카시 포트 드라이버는 또한 미니포트 드라이버를 초기화를 담당한다. 스카시 미니포트 드라이버(miniport drivers: MPDs)는 스카시 디바이스에 대한 하드웨어 드라이버이다. 그것들은 인터럽트와 위로부터 요청을 수행하는 입출력 포트-레벨 상호작용을

처리한다. 그것들은 또한 어댑터-특정 에러복구를 수행할 수 있다.

포트 드라이버(port drivers: PDRs)(비-스카시 하드웨어에 대하여)는 스카시 포트와 미니포트 드라이버로서 유사한 기능을 수행한다. 그것들은 입출력을 수행하기 위하여 하드웨어와 직접적으로 상호작용하는 32-비트 디스크 엑세스를 제공한다. 실제 모드 맵퍼(real mode mapper: RMM)는 윈도우즈 95가 포트 드라이브를 제공할 수 없는 어떤 상황에서 사용된다. 플러그앤플래이(plug-and-play) BIDS의 소개와 함께, 많은 하드웨어 특정 포트 드라이버를 포함하므로서, 윈도우즈 95는 일반적으로 대부분의 디스크 하드웨어에 대한 32-비트 엑세스를 제공할 수 있다. 그러나 윈도우즈 95는 소수에게만 보유된 하드웨어 함께 오래된 개인용 컴퓨터에 작동되어야 했다. 그래서, 보호 모드에서 디스크 입출력을 처리하기 위한 포트 드라이버를 제공할 수 없는 그 경우에 대해 허용을 해야 한다. 시스템은 또한 보호 모드에서 디스크 입출력을 처리하기 위한 실제-모드 디스크 드라이버 소프트웨어를 사용한다. 이러한 상황에 대해, 보호 모드 가상 디바이스 드라이버의 체인에서 최근의 엔트리(entry)는 포트 드라이버 대신에 실제 모드 맵퍼(RMM)이다. 실제 모드 맵퍼는 하드웨어 입출력을 수행하기 위하여 실제 모드 드라이버를 호출하고 리턴은 상기 파일시스템 체인으로 귀착된다. 실제 모드 드라이버는 특정 시스템의 하드웨어나 소프트웨어 구성으로 요구되는 하드웨어 드라이버이다. 그러나, 실행이 손상을 입을 수 있기 때문에 실제 모드 드라이버의 이용은 방해된다(실제 모드에서 보호로부터 이행과 실제 모드에서 느린 실행의 오버헤드(overhead)에 기인하여). 그러나, 유연성과 뒤로의 호환성에 대해 그것들을 승인한다.

일반적으로, 파일시스템 구조의 상위 층은 윈도우즈 95의 부분으로서 마이크로소프트에 의해 작성되었다. 반면에, 하위 층은 디스크 드라이브 제조업자에 의해서 제공된다. 결론적으로, 써드-부(third-party) 개발자에 의한 개발을 위해 일반적으로 사용되는 층은 상기 가상 공급 드라이버 층이다. 전술한 바와 같이, 본 발명에 따라서, 어떤 은폐 파일이 사용자로부터 숨겨지는 것을 보증하기 위해서, 가상 공급 드라이버는 파일시스템 호출을 차단하고 호출 프로세싱을 수행하도록 사용될 수 있다.

본 발명의 일 실시예에서, 25 벤더 공급 드라이버는 사용자로부터 은폐 파일을 숨기기 위하여 파일시스템 " 열기, " 처음 찾기" , " 다음 찾기" , " 삭제" , " 이름바꾸기" 의 호출을 차단하도록 사용된다. 이러한 호출의 차단은 도 23의 단계 2210)에서 일어난다. 위의 호출은 단계 2212에서 그때 확인된다. 그래서 호출 프로세싱은 단계 2216에 의하여 수행될 것이다.

도 26A-26G는 호출과 " 열기" , " 처음 찾기" , " 다음 찾기" , " 삭제" , " 이름바꾸기" 호출의 리턴 프로세싱하도록 적용할 수 있는 프로세싱과 호출 차단의 일 실시예의 플로우차트이다. 파일시스템 호출 차단은 단계 2510에서 행해진다. 호출 차단 이후에, 상기 호출은 그것이 중요한 호출의 타입인지를 확인하기 위하여 평가된다. 이러한 평가는 단계 2512에서 단계 2520을 통하여 수행된다. 단계 2512에서, 은폐 드라이버는 상기 호출이 " 열기" 호출인지를 결정한다. ; 단계 2514에서 상기 호출이 " 처음 찾기" 호출인지를; 단계 2516에서 상기 호출이 " 다음 찾기" 호출인지를; 단계 2518에서 상기 호출이 " 삭제" 호출인지를; 그리고 단계 2520에서 상기 호출이 " 이름바꾸기" 호출인지를 결정한다. 어떤 " 아니오" 결정은 플로우차트 단계 2512-2520로 이동한다. 호출의 어떤 것도 중요한 타입이 아니라면, 상기 호출은 단계 2522A에서 다음 층으로 향하게 되고, 그러므로 상기 호출은 은폐 드라이버(2118)에 의해 변경없이 통과된다.

단계 2512에서, 상기 호출은 " 열기" 호출인 것이 결정되고, 상기 파일이름은 상기 파일이 은폐될지를 결정하기 위해 조사된다. 은폐 드라이버(18)는 상기 파일이 이전에 은폐되었는지를 확인하기 위하여 룩업(lookup) 테이블 또는 데이터베이스에서 파일이름을 찾는다. 상기 파일이 은폐되지 않는다면, 상기 은폐 드라이버(18)는 단계 2522A에서상기 호출을 다음 층으로 보낸다. 상기 파일이 은폐된다면, 상기 은폐 드라이버(2118)는 단계 2532에서 리턴 변수를 발생시키며, 파일시스템 관리자를 통과하고, 마침내 파일이 없거나 어떤 에러가 있다는 것을 사용자에게 알린다. 예를 들어, 은폐 드라이버(2118)는 " 파일 없음" 메세지를 나타내는 값에서 에러 코드를 설정할 수 있고 호출 실행의 실패를 암시

하기 위해 리턴 변수로 " -1" 을 설정한다. 추가적인 보호수단으로, 은폐 드라이버(2118)는 파일 처리를 처리가 존재하지 않는다는 것을 설치가능 파일시스템 관리자에 나타내는 널(NULL)로 설정한다. 단계 2532의 최종결과는 파일이 열리는 것을 막는 것 뿐만아니라, 파일이 시스템에 존재하지 않는 것을 사용자에게 알리기위한 것이다. 그러므로, 사용자가 어떤 파일(예를 들어, " SECRET.doc")이 시스템에 존재하는지 의심을 가진다고 가정하면, " 열기" 명령을 이용하여 파일 엑세스를 위한 시도는 단계 2532에 의하여 널(null) 처리와 에러 메세지를 리턴하므로서 좌절된다.

단계 2514는 " 처음 찾기" 호출을 검사한다. 이 호출이 찾아지면, 이것은 시스템 층 아래로 그리고 리턴 위로 통과되며, 복구된 파일이름은 상기 요청이 은폐 파일을 목적으로 했는지를 단계 2542에서 조사된다. 그것이 은폐되지 않는다면, 드라이버는 단계 2522B를 진행하고 설치가능 파일시스템 관리자에서 상기 호출을 직접적으로 통과시킨다. 단계 2542에서 상기 파일이 은폐 파일이라면, 은폐 드라이버(2118)는 단계 2544를 진행하고 " 다음 찾기" 기능을 실행시킨다. 단계 2546에서 리턴되어, 상기 리턴 변수는 상기 타겟 파일이 은폐되는지를 다시 확인하기 위해 조사된다. 그것이 은폐되지 않는다면, 프로그램은 단계 2522B를 진행한다. 타겟 파일이 은폐 되면, 또 다른 " 다음 찾기" 기능을 다시 실행시키기 위해서 프로그램 루프(loop)는 단계 2544로 돌아간다. 이러한 방법으로, 이름과 은폐 파일의 존재는 사용자로부터 계속 숨겨지게 되고, 처음 찾은 은폐되지 않은 파일만이 사용자에게 보여질 것이다.

전술된 유사한 프로시저(procedure)는 " 다음 찾기" 호출에 대해 수행된다. 그러므로, " 다음 찾기" 호출이 단계 2516에서 확인되면, 상기 호출은 파일이름을 리턴하기 위하여 단계 2550에서 상기 체인 아래로 통과된다. 그리고 단계 2552에서 조사된 이름이 은폐 파일과 일치하지 않는다면 다른 " 다음 찾기" 호출은 설치가능 파일시스템 매니저에서 첫번째 파일이름의 리턴을 허락하지 것 없이 단계 2554에서 상기 체인 아래로 통과된다. 이 " 다음 찾기" 기능의 리턴이 단계 2556에서 조사되고, 그것이 은폐 파일을 가리키지 않는다면, 프로그램은 설치가능 파일시스템 매니저에서 리턴을 통과시키기 위해 단계 2522B를 진행한다. 상기 리턴이 단계 2556에서 은폐 파일을 확인한다면, 프로그램은 또 다른 " 다음 찾기" 기능을 실행하기 위해 단계 2554로 다시 돌아간다. 이런 방법으로, 다음 찾기 기능은 은폐 파일의 확인을 운영체제에서 확인하지 않는다.

" 처음 찾기" 와 " 다음 찾기" 호출은 일반적으로 디렉토리를 목록으로 나타내는데 사용되고, 그러므로 플로우차트에 기술된 프로시저는 그것이 디렉토리 리스트에서 나타내지는 것을 방지하므로서 은폐 파일을 효과적으로 숨긴다.

단계 2518은 " 삭제" 호출을 조사한다. 이 호출이 찾아지면, 타겟 파일이 은폐되는지 않되는지가 단계 2560에서 결정된다. 상기 파일이 은폐되지 않는다면, 단계 2552A는 상기 호출이 다음 층 아래로 통과되도록 수행된다. 타겟 파일이 은폐되면, 에러 코드를 2로, 리턴 코드를 실패를 나타내는 -1로 세팅하므로서 " 파일 없음" 이라는 에러 메세지가 출력되고 파일시스템 매니저로 리턴된다. 이 방법으로, 사용자는 타겟 파일을 삭제할 수 없으며, 실제로 시스템은 상기 파일이 마치 존재하지 않는 것처럼 작동한다 - 정확히 파일을 은폐시키기 위해 요구되는 결과이다.

단계 2520에서, 상기 " 이름바꾸기" 호출은 조사된다. 이 호출이 찾아진다면, 은폐 드라이버(2118)는 타겟 파일이 은폐되는지를 결정하기 위하여 단계 2570를 진행한다. 상기 파일이 은폐되지 않는다면, 상기 호출은 단계 2552A에서 다음 층 아래로 통과된다. 상기 파일이 은폐된다면, 리턴 변수는 단계 2562에서 한 것처럼 에러 코드를 2로 설정하고 -1을 리턴하도록 단계 M에서 발생된다.

은폐 작동은 사용자가 " 열기" , " 처음 찾기" , " 다음 찾기" , " 삭제" , " 이름바꾸기" 호출 중 하나로 결정되는 프로시저를 초기화할 때마다 실제-시간(real-time)에 실행된다는 것을 이해할 수 있다.

파일 호출의 감시와 위에서 나타낸 리턴은 은폐 드라이버(2118)에 의해 수행된다. 일반적으로, 상기 파일과 리턴은 설치가능 파일시스템 관리자로부터 하위 레벨 드라이버 아래로 진행되는 요청을 가진 " 설치가능 파일시스템 요청 패킷" 의 부분이고, 상기 리턴과 리턴 변수는 상기 패킷에 삽입되고, 하위 레벨 드라이버로부터 설치가능 파일시스템 관리자로 상기 체인을 보낸다.

윈도우즈 NT 환경에서 도 26A-26G에 나타낸 원리를 확장하는 것은 상기 " 열기" , " 삭제" , " 이름바꾸기" 호출에 대하여 간단하다. 그러나, " 처음 찾기" 와 " 다음 찾기" 에 대해서, 그 이상의 프로세싱이 디렉토리에서 파일 리스트를 나타내도록 리턴되는 인덱스 버퍼(index buffers)를 처리하기 위해서 요구된다. NTFS 파일시스템은 처리 데이터베이스 방법(transaction database maner)으로 작동하고 빠른 룩업(look-up)과 목적하는 것을 출력하기 위해 그 파일에 색인을 단다. 그런 인텍스는 각 부분에 대해서 맞춰진다. 그러므로, 파일을 은폐시키기 위해서, 상기 파일은 메모리부로부터 제거되어야 하고 이때 인덱스 진행은 은폐 드라이버에 리턴된다.

도 26A-26G에 도시된 본 발명의 실시예는 본래 운영체제 사용자가 은폐된 어떤 파일의 존재를 알게되는 것을 막는 것이다. 그러나, 어떤 애플리케이션에 있어서, 디렉토리에서 사용자가 은폐 파일을 목록에 올리도록 허가하는 것이 바람직할 수 있으나 다른 점에서 상기 파일을 엑세스할 수 없다. 상기 파일은 엑세스-방지된다. 본질적으로, 사용자는 단지 사용자가 디렉토리 리스트에서 파일이름을 보므로서 상기 파일의 존재를 알 수 있다는 점에서 상기 파일의 유효한 엑세스가 금지된다. 이러한 실시예에서, 디렉토리 리스팅(listing)은 보호된 암호가 없으나, 파일을 보는 것 이상의 엑세스(access)는 단지 엑세스 권한을 가진 자만이 상기 파일을 엑세스할 수 있도록 허가하기 위한 보호된 암호이다. 이러한 실시예에서 상기 은폐 파일은 파일 내용이 사용될 수 없도록 보증하기 위해 바람직하게 암호화된다. 그러한 일 실시예는 기밀의 파일이 조작자에 의해 엑세스될 거라는 걱정 없이, 어떤 자가 기밀의 파일을 찾는 컴퓨터나 메모리 디바이스를 숨기도록 허가하는 애플리케이션을 가진다.

본 발명의 위 실시예의 실행은 단지 단계 2514와 단계 2516에서 각각 열기 찾기와 다음 찾기 호출을 제거하도록 요구한다. 이러한 방법에서 디렉토리 리스팅은 열기, 삭제를 제외하고 가능할 것이고, 이름바꾸기 프로세스는 허가되지 않을 것이다.

이미 본 발명의 또 다른 실시예는 해당 프로세스 또는 애플리케이션 그리고 상기 애플리케이션 내에 있는 선택된 프로시저에 적용할 수 있는 승인과 제한을 이용한다. 상기 파일은 사용자-제한이라 한다. 예를 들어, 사용자는 워드프로세스 프로그램이 파일을 읽고 편집하고 파일을 복사하는 것을 허가하는 본 발명의 이러한 실시예를 이용할 수 있다. 그러나 파일을 삭제하는 것은 허가하지 않는다.

이 실시예에서, 도 26A-26G의 감시 시스템은 단계 2518, 2560, 2552A, 2562에서 삭제 호출 감시를 제외한 모든 호출 감시를 제거하기 위해 변경된다. 추가된 논리적 질문은 감시 호출 또는 요청 패킷이 워드(Word)와 같이 자명한 워드프로세스 애플리케이션을 포함할지를 결정한다. 은폐 드라이버가 요청 애플리케이션이 워드였다고 결정되고, 삭제 호출이 상기 파일시스템 매니저로부터 보내졌다면, 그때 결과는 삭제 감시 단계 2518, 2560, 2552A, 2562에서 갈라진다. 이 방법에서, 본 발명의 실시예는 파일을 은폐하지 않는다. 그러나 해당 애플리케이션에서 보통 사용할 수 있는 적어도 한 부분에서 파일의 사용을 제한한다.

은폐 애플리케이션(2116)은 사용자가 파일이 데이터베이스 또는 은폐 드라이버(2118)의 룩업 테이블에 저장(삭제)되는 것을 선택하는 것(또는 선택하지 않는 것)을 허락한다. 은폐 애플리케이션(2116)은 또한 은폐되고, 엑세스 금지되고, 사용자 제한된 작동 모드에 적용할 수 있는 승인의 다른 타입으로 사용자 암호, 암호화 선택에 대해 사용자 인터페이스를 나타낸다. 은폐 애플리케이션은 초기에 선택된 디렉토리에서 모든 파일을 목록에 올리고 사용자에게 은폐되도록 요구되는 가장 중요한 그 파일을 허가한다. 상기 가장 중요한 점은 효과적으로 은폐된 파일을 선택(다이알로그상자

에서 " O.K." 명령 또는 엔터를 누름)하므로서, 사용자는 선택된 파일이 은폐 드라이버 데이터베이스 또는 룩업 테이블에 저장되도록 한다. 사용자에게는 바람직하게 사용자가 더 최근에 파일의 은폐를 풀거나 은폐된 새 파일을 추가할 수 있도록 암호를 입력하도록 요구된다. 암호는 초기에 입력될 수 있으나 파일 선택후에 교대로 입력될 수도 있다. 그러나 어떤 부분에서는 파일시스템 매니저로부터 상기 호출을 감시하는 드라이버 프로그램 실행 전에 입력될 수 있다. 파일의 은폐를 풀기 위해서, 사용자는 은폐 애플리케이션으로부터 승인된 자신의 암호를 재입력하며, 그때 은폐 드라이버 데이터베이스(또는 룩업 테이블)에 목록화 되어있는 파일이름을 추출해서, 보통의 리스트 디렉토리 명령 동안 바람직하게 출력될 수 있다. 은폐된 파일은 가장 중요하거나 또는 다른 방법으로 디렉토리 리스팅에서 구별되서, 사용자는 데이터베이스나 룩업 테이블로부터 그것들을 삭제 그리고 그것들을 은폐에서 풀기를 선택에서 해제하는 것이 어떤 파일인지를 알게될 것이다. 같은 시간에 다른 파일은 은폐, 엑세스 금지 그리고/또는 사용자-제한을 위해 선택될 수 있다.

은폐, 엑세스 금지, 사용자-제한의 모드는 은폐 애플리케이션의 그래픽사용자인터페이스(GUI)로부터 다이알로그상자 또는 다른 입력을 이력하여 각 파일에 대해서 선택될 수 있다. 프로그램 흐름(flow)은 파일 기반으로 각각의 모드를 실행시킨다. 그러므로 파일 A와 B는 은폐될 수 있고, 파일은 엑세스 금지될 수 있으며, 파일 D와 E 사용은 제한된다. 모드/파일 조합의 초기 선택은 애플리케이션 프로그램을 이용하여 파일/모드 선택 프로세스가 진행되는 동안 사용자에 의해 이루어진다. 또 다른 작동 모드에서, 숨은 파일은 운영체제의 사용자에 의해 선택될 필요가 없으나, 운영체제에 설치를 위해 애플리케이션을 판매하는 써드-부(third-party) 벤더에 의해 미리-선택될 수 있다. 써드부 벤더는 운영체제의 사용자에 의해 우연히 삭제되는 것으로부터 어떠 파일이나 .exe 파일을 유지하고자 하고 미리 상기 애플리케이션에 의한 엑세스를 위해 이용가능한 이 파일을 가진다. 예를 들어 애플리케이션 X는 사용자가 은폐되고, 등등, 운영체제에서 볼 수 없도록 바라는 파일 Y와 Z를 포함하며, 애플리케이션 X가 엑세스를 바라는 것은 예외이다. 이 경우에서, 애플리케이션 X는 애플리케이션 X를 제외한 운영체제로부터 파일 Y와 Z를 은폐시키는 은폐 드라이버를 포함한다. 이 작동은 본질적으로 파일 Y와 파일 Z의 사용자-제한이다. 애플리케이션 X가 파일 Y와 Z를 엑세스하려 하면, 운영체제는 파일 매니저를 통해 완전한 엑세스와 사용자 승인을 제공할 수 있다. 사용자가 애플리케이션 외부의 엑세스를 하고자 하면, 그런 작동은 취소되고 운영체제는 파일이 마치 존재하지 않는 것(찾을 수 없는 것)처럼 작동할 것이다. 사용자에 의해 파일의 우연한 삭제를 방지하는 애플리케이션은 사용자에 의해 파일 복사를 방지하는 것이다. 그런 애플리케이션은 인터넷상에 분포된 오디오와 비디오 내용의 파일과 같은 중요한 애플리케이션이다. 이 경우에, 애플리케이션 A는 애플리케이션 A가 승인을 이용하기 때문에 은폐 파일을 " 작동" 시킬 수 있다. 그러나, 사용자는 운영체제의 파일시스템 매니저를 통하여 파일을 처리(이동, 복사, 이름바꾸기, 삭제, 디렉토리 목록)할 수 없다.

물론, 은폐 드라이버는 애플리케이션 A를 통한 것외에 사용자에게 어떤 사용 승인을 주도록 이용될 수 있으며, 예를 들어, 상기 리스트 디렉토리와 삭제 명령은 단지 인터넷상에서 파일의 복사 또는 재전송을 방지하는 다른 명령을 차단한다.

본 발명의 여러가지 실시예의 은폐 드라이브는 인터넷을 통하여 사용자에게 다운로드되거나 CD를 통하여 사용자에게 제공된다. 다운로드 상황에서, 프로그램은 서버 메모리에 저장되고 전자상거래(e-commerce) 방식으로 다운로드된다.

상술된 바와 같이, 파일 은폐 또는 숨기기는 파일이 디렉토리 리스팅에 목록으로 오르는 것을 차단하고 컴퓨터 디바이스의 운영체제가 파일 엑세스하는 것을 차단한다. 그러므로, 상기 파일은 컴퓨터 운영체제의 파일시스템으로부터 열기, 삭제, 이름바꾸기, 엑세스될 수 없다. 적어도 호출과 리턴은 파일이름과 같은 특정 파일 확인 데이터와 관련된다는 점에서, 은폐 드라이버는 호출을 확인하고 파일 매니저에 리턴한다. 이 관계는 상기 호출 또는 리턴 패킷 또는 포인터(pointers)에 대한 간접적인 참조에서 이름에 대해 직접적인 참조가 될 수 있으며 상기 파일을 유일하게 확인하기 위해서 파일이름 또는 확인 데이터를 차례로 확인한다. 이 방법에서, 취소되는 작동(열기, 처음 찾기, 다음 찾기, 삭제, 이름바꾸기)은 유일하게 특정 파일과 관계될 수 있으며 그 확인은 은폐, 엑세스-금지 또는 사용-제한되도록 선택되는 파일에만 취소를 적용하기 위하여 은폐 드라이버 데이터베이스나 룩업 테이블에 대해 검사될 수 있다.

다음은 본 발명에 따른 패키지의 바람직한 실시예를 설명한다. 이미 나타낸 바와 같이, 패키저(12)는 바람직하게 클라이언트 또는 리시버(14)와 결합하여 작동한다.

현재 기술로서 알려진 문제점은 전자 메세지의 저자는 그것이 네트워크를 통하여 전송된 후에 상기 메세지에 어떠 일이 일어나도록 제어할 수 없다는 것이다. 예를 들어, 수신자는 상기 메세지를 다른 사용자에게 전송, 상기 메세지를 프린트, 메세지를 확인 후 저장, 상기 메세지를 클립보드에 복사할 수 있다. 저자는 써드-부(third party)에 전해지는 기밀의 전자우편이나 메세지 또는 미래의 참고를 위해 저장되거나 프린터된 메세지의 복사를 원하지 않는다.

어떤 전자우편 프로그램은 저자가 " 개인용" 메세지를 나타내도록 허가한다. 이러한 설정은 수신자가 초기의 메세지를 변경하는 능력을 제한하고 상기 메세지가 상기 저자로부터 전송되는 것을 나타내며 그것을 써드-부로 전송한다. 그러나, 이러한 설정은 최초의 형태에서 전송, 클립보드에 복사, 상기 메세지를 저장 또는 프린트하는 능력을 사용자에게 제한하지는 않는다.

전송된 정보를 사용하는 수신자의 능력을 제한하는 통신상에서 저자가 승인을 설정하도록 허가하는 방법과 장치를 가지는 전자와 디지털통신 분야에서 필요성이 존재한다. 게다가, 상기 저자가 단지 의도한 수신자만이 상기 메세지를 받는 것을 보증하도록 허가하는 방법과 장치에 대한 필요성이 있다. 본 발명은 데이터 파일, 유일한 식별자, 상기 파일의 사용을 통제하는 하나 이상의 승인으로 이루어진 데이터의 암호화된 패키지를 생성시키는데 대하여 방법과 장치를 제공하므로서 이러한 필요성을 제시한다. 상기 패키지는 또한 수신자의 유일한 식별자와 상기 패키지 수신에서 상기 수신자의 컴퓨터 시스템에 설치되는 클라이언트 소프트웨어 패키지를 포함할 수 있다.

본 발명의 다른 방법은 상기된 방법에 운영체제가 탑제된 클라이언트 컴퓨터 시스템에서 상기 컴퓨터 실행파일을 수신하는 단계와 상기 클라이언트 컴퓨터 시스템에서 상기 컴퓨터 실행파일을 실행하는 단계를 추가한다. 상기 파일의 실행 단계는 상기 운영체제가 호환성있는 운영체제인지를 결정하는 단계와, 그렇지 않다면 상기 클라이언트 컴퓨터 시스템에 클라이언트 소프트웨어를 실행하는 단계로 구성된다. 상기 클라이언트 소프트웨어의 실행은 클라이언트 승인 데이터베이스와 상기 클라이언트 컴퓨터 시스템에 저장소를 생성시킨다. 상기 클라이언트 소프트웨어 실행 후에,상기 방법은 상기 암호화된 패키지가 유효한지를 결정하는 단계와, 그렇다면 상기 패키지 전체의 유일한 식별자를 상기 클라이언트 승인 데이터베이스에 기록하는 단계, 데데이터의 상기 패키지로부터 상기 데이터 파일과 하나 이상의 승인을 추출하는 단계, 상기 데이터 파일을 상기 저장소에 저장하는 단계, 상기 하나 이상의 승인을 상기 클라이언트 승인 데이터베이스에 저장하는 단계를 더 포함하여 구성된다. 상기 패키지가 유효하지 않다면, 상기 방법은 상기 패키지가 설치되는 것을 나타내기 위해서 상기 컴퓨터 실행파일에서 상태를 설정한다.

도 27은 본 발명의 바람직한 실시예에 따라 정보의 패키지를 통신하는 시스템의 블록다이어그램을 도시한다. 패키저(2612)는 " package.exe" (2614)" 와 같은 실행 파일을 생성하여 클라이언트가 접근할 수 있도록 네트워크(2616)상에서 클라이언트 컴퓨터 시스템(2617)에 전송한다. 상기 컴퓨터 실행파일(2614)은 패키저(2612)에 의하여 수집된 정보의 패키지를 포함한다.

본 발명의 일 실시예에 따르면, 상기 정보의 패키지는 데이터 파일(2618)과 승인 데이터베이스(2620)를 포함한다. 본 발명의 다른 실시예에서는 상기 정보의 패키지는 암호화 소프트(2622)와 필수적이지는 않지만 선택적으로는 클라이언트 소프트웨어(2624)를 포함한다. 바람직하게 상기 클라이언트 소프트웨어는 버전 데지그네이션(version designation)을 갖는다. 패키저(2612)는 각각의 정보 패키지에 대하여 하나의 패키지 전역의 유일한 식별자(a package global unique identifier: PGUID)를 생성하며, 그것을 정보의 패키지에 포함시킨다. 바람직한 실시예에서, PGUID를 포함하는 정보의 패키지는 암호화 소프트웨어(2622)에 의하여 암호화 된다. 예를 들어 상기 PGUID는 일련의 수문자식 심벌일 수 있다.

본 발명의 방법에 따르면, 패키저(2612)는 데이터 파일(2618)과 승인 데이터베이스(2620)를 수신한다. 상기 승인 데이터베이스(2620)는 상기 데이터 파일(2618)과 연계하여 상기 데이터 파일(2618)의 사용을 통제하는 하나 또는 그

이상의 저자-구성 승인(author-configurable permissions)을 갖는다. 이러한 저자-구성 승인의 한가지 기능은 데이터 파일(2618)의 공유를 못하게 하는 것이다. 예를 들어 저자-구성 승인은 엑세스 회수, 접근 시간, 소멸 기일, 권한 부여 기일, 클립보드 승인, 프린트 승인, 무제한적 접근 승인, 애플리케이션 승인, 및 시스템-이벤트 승인을 포함한다.

엑세스 회수 승인은 사용자가 데이터 파일(2618)에 엑세스하도록 허용되는 시기의 수를 구체화 한다. 일 실시예에서, 한번의 엑세스 회수는 클라이언트 컴퓨터 시스템(2617)상에서 하나의 프로세스가 그 프로세스의 수명을 위하여 상기 데이터 파일(2618)에 엑세스하도록 허용하는 것으로서 정의된다. 상기 엑세스 시간 승인은 클라이언트가 파일에 접근할 수 있는 시간의 총량을 구체화한다. 클라이언트 컴퓨터 시스템(2617)상에서 어떤 프로세스가 상기 데이터 파일(2618)을 열면, 그 엑세스접근 시간은 그 프로세스가 끝날 때 까지 감소되거나 또는 그 프로세스의 종료 이전에 엑세스 시산이 완전히 소진되면 그 프로세스는 자동으로 종료된다.

만료 날짜 승인은 상기 데이터 파일이 더 이상 엑세스될 수 없는 날짜를 구체화한다. 만료 날짜가 도래할 때까지 클라이언트는 데이터 파일(2618)상의 다른 승인들을 조건으로 그 파일에 무제한적으로 엑세스할 수 있을 것이다. 만일 클라이언트 컴퓨터 시스템(2617)상에서 어떤 프로세스가 만료 날짜에 상기 데이터 파일(2618)을 열게 한다면, 그 프로세스는 자동적으로 종료된다. 만료 기일에 상기 파일 내용은 덮어쓰기되거나(overwritten) 또는 삭제된다. 바람직하게, 상기 만료 날짜 승인은 또한 승인 데이터베이스로부터 제외된다.

상기 권한부여 날짜 승인은 상기 데이타 파일(2618)을 엑세스할 수 있게 되는 날짜를 구체화한다. 데이터 파일(2618)상의 다른 승인들을 조건으로, 사용자는 그 날짜가 경과 될 때 까지 상기 데이터 파일(2618)을 엑세스할 수 없다. 이러한 모든 엑세스 승인들은 개별적으로 또는 조합으로 구성되고 실시될 수 있다.

상기 클립보드 승인은 클라이언트가 상기 데이터 파일(2618) 또는 그 파일의 일부를 예를 들어 윈도우즈 클립보드등에 복사할 수 있을 지를 구체화한다. 상기 클립보드 승인은 또한 상기 테이타 파일을 다른 컴퓨터 시스템에 전송하는 것을 방지하도록 구성될 수 있다.

상기 프린터 승인은 클라이언트가 상기 데이터 파일(2618)을 프린트할 수 있을지를 구체화한다. 상기 무제한적 엑세스 승인은 클라이언트가 상기 데이터 파일(2618)에 무제한적으로 접근하는 것을 허용한다. 바람직하게, 상기 데이터 파일(2618)은 읽기 전용이며, 클라이언트는 무제한적 엑세스 승인으로 시간 제한 없이 데이터 파일(2618)을 볼 수 있다. 그러나 상기 클라이언트는 다른 승인들 예를들어 프린트 승인 및 클립보드 승인등이 상기 데이터 파일(2618)에 연계되지 않는 한 더 이상이 것이 허용되지 않을 것이다.

상기 애플리케이션 승인은 상기 클라이언트 컴퓨터 시스템(2617)상에서 애플리케이션 목록 중의 하나 또는 그 이상이 실행 중인지를 결정하고 만약 애플리케이션중의 하나가 실행 중인 경우에는 데이터 파일을 엑세스할 수 없게 한다. 선택적으로, 상기 애플리케이션 승인은 만약 상기 응용 중의 하나가 실행중이 아닌 경우에는 상기 데이터 파일에 접속할 수 없게 할 수 있다. 상기 시스템-이벤트 승인은 상기 클라이언트 컴퓨터 시스템(2617)을 분석하여 시스템-이벤트가 발생하는지를 결정하고 발생한 시스템-이벤트를 근거로 상기 데이터 파일(2618)을 엑세스하도록 허가할 것인 지를 결정한다.

바람직한 실시예에서, 상기 패키저(2612)는 상기 패키지에의 접근을 제한하는 패스워드를 한정할 수 있다. 상기 정보의 패키지는 클라이언트가 클라이언트 컴퓨터 시스템(2617)에 적절한 패스워드를 입력할 때 까지 접근될 수 없을 것

이다. 다른 실시예에서, 상기 패키저(2612)는 수신자 전역의 유일한 식별자(recipient global unique identifier: RGUID)를 받아 그것을 정보의 패키지 내에 포함시킬 수 있다. 상기 RGUID는 저자가 상기 데이타 파일(2618)을 전송하고자하는 클라이언트를 확인하는 것으로 저자에 의해 수동으로 패키지 내에 입력되거나 상기 패키저(2612)에 저장된 클라이언트 목록으로부터 선택될 수 있다.

정보의 패키지는 컴퓨터 실행 파일(2614)에 연계되어 네트워크(2616) 상에서 상기 클라이언트 컴퓨터 시스템(2617)에 전송된다.

도 28A-28B는 본 발명의 일 실시예에 따라, 컴퓨터 실행 파일이 생성된 후 정보의 패키지를 통신하는 방법을 도시한다. 이 방법에 따르면, 상기 컴퓨터 실행 파일(2714)은 그 방법을 실행하는 코드로 이루어지며, 상기 클라이언트 컴퓨터 시스템(2717)에서 실행된다(단계 2730). 상기 패키지가 암호로 보호되는 실시예에서, 클라이언트는 암호를 입력하게 될 것이다. 단계 2732에서, 상기 클라이언트 컴퓨터 시스템(2717)은 그 운영체계가 적합한 운영체계인 지를 결정한다. 만약 그 운용체계가 적합하지 않다면, 상기 패키지는 삭제 및 덮어쓰기된다(단계 2734). 적절한 운용체계는 제한적이지 않으나 윈도우즈 95, 98, NT 및 2000을 포함한다. 선택적으로는, 도 28의 단계 2736에서와 같이, 상기 클라이언트 컴퓨터 시스템(2717)은 정보의 두 번째 패키지가 상기 클라이언트 컴퓨터 시스템(2717)에 이미 탑재되었는지를 결정하고, 만약 그렇다면, 상기 두 번째 패키지를 종결한다(단계 2738). 단계 2740에서, 상기 시스템(2717)은 상기 클라이언트 소프트웨어(2724)가 설치되었는지를 결정한다. 만약 클라이언트 소프트웨어가 설치되지 않았다면, 상기 클라이언트 소프트웨어(2724)는 상기 패키지로부터 인출되어 상기 클라이언트 컴퓨터 시스템(2717)에 설치된다(단계 2742). 만약 상기 클라이언트 소프트웨어가 설치되었다면, 상기 시스템(2717)은 상기 패키지내 상기 클라이언트 소프트웨어의 버전과 설치된 소프트웨어의 버전을 비교한다(단계 2744). 만약 상기 패키지내 클라이언트 소프트웨어의 버전이 상기 시스템(2717)상에 설치된 버전보다 늦다면, 상기 패키지로부터 보다 새로운 버전을 추출하고 그것을 상기 시스템(2717)상에 설치함으로써 그 설치된 클라이언트 소프트웨어는 업그레이드된다(단계 2742). 다른 실시예에서, 클라이언트 소프트웨어(2724)는 상기 패키지로부터 추출되고 설치된 버전을 체크하지 않고 설치될 수 있다.

단계 2748에서, 클라이언트 소프트웨어(2748)가 실행되어 상기 클라이언트 컴퓨터 시스템 상에 클라이언트 승인 데이터베이스 및 저장소를 생성한다. 상기 저장소는 운영체계에 완전히 통합된 가상의 디스크 환경이지만 운영체계로부터 분리되어 고유의 운영 규칙이 실행될 수 있으며 상기 시스템에 전혀 위험을 주지 않고 그 안에서 상기 데이터 파일을 검사할 수 있다. 도 28에 도시된 실시예에서, 클라이언트 소프트웨어(2724)는 상기 소프트웨어의 실행에 따라 클라이언트 컴퓨터 시스템(2717)상에 설치되는 하나 또는 그 이상의 디바이스 드라이버 및 하나 또는 그 이상의 WIN32 모듈을 포함한다(단계 2750). 장치 드라이버 또는 WIN32 모듈 중 적어도 하나는 상기 클라이언트 승인 데이터베이스 및 저장소를 생성한다(단계 2752). 바람직한 실시예에서, 상기 WIN32 또는 장치 드라이버는 수정된 WIN32 실행 드라이버이다. 상기 장치 드라이버들과 WIN32 모듈들은 또한 소프트웨어(2724)의 또 다른 기능, 예를 들어 상기 승인 구조가 변경되어 있는 지를 확인하는 등의 기능을 수행한다. 모든 장치 드라이버들과 WIN32 모듈들이 일단 탑재되면 그것들은 저장소 내에서 해킹을 방지하는 역할을 한다(단계 2754). 상기 클라이언트 소프트웨어가 설치된 후에는, 상기 시스템(2717)에 전원이 공급될 때 상기 수정된 WIN32 실행 드라이버가 초기화되도록 운영체계가 수정된다.

상기 장치 드라이버 중 적어도 하나는 컴퓨터 실행 파일(2714)과 통신한다. 일 실시예에서, 상기 WIN32 모듈 중의 하나는 정보의 패키지를 조회하기 위한 요구를 수신한다(단계 2756). 그러면 상기 WIN32 모듈은 상기 정보의 패키지가 암호로 보호되어 있는 지를 결정하고(단계 2758), 만약 그렇다면 클라이언트에게 암호를 묻는다(단계2760). 만약 상기 패키지가 암호로 보호되지 않거나, 또는 암호로 보호되고 정확한 암호가 입력되면(단계 2762), 상기 WIN32 모듈은 상기 패키지가 유효한 지를 결정한다(단계 2764). 만일 상기 패키지가 유효하다면, 상기 데이터 파일은 저장소 내로 흡수되고, 하나 또는 그 이상의 승인은 상기 클라이언트 승인 데이터베이스에 저장되고, 상기 클라이언트 승인 데이터베이스는 PGUID로 업데이트된다(단계 2766).

바람직하게, 상기 패키지의 유효성은 상기 패키지로부터 상기 PGUID를 판독하고 PGUID에 대한 상기 클라이언트 승인 데이터베이스를 체크함으로써 결정된다. 만약 PGUID가 클라이언트 승인 데이터베이스 내에 있다면, 패키지는 이미 다른 시간에 저장소 내로 수신되었으며 상기 패키지는 실효가 없다. 이와 같이 발생한다면, 상기 패키지가 이미 설치되어 있음을 지시하기 위하여 컴퓨터 실행파일(2714) 내의 상태(state)가 설정된다. 상기 상태를 설정하는 것은 예를 들어, 데이터 비트를 변경하거나 플래그(flag)를 설정하는 것이다. 만약 상기 PGUID가 상기 클라이언트 승인 데이터베이스에 없다면, 상기 패키지는 저장소에 새로 추가되고 그것은 유효하다.

상기 정보의 패키지 내에 RGUID를 갖는 실시예에서, 패키지의 유효성은 RGUID에 대한 클라이언트 승인 데이터베이스를 체크함으로써 결정된다. 만약 RGUID가 클라이언트 승인 데이터베이스 내에 있다면, 상기 패키지는 클라이언트가 상기 패키지를 수신하도록 될 것이며, 그 패키지는 유효하다. 만약 RGUID가 클라이언트 승인 데이터베이스 내에 존재하지 않는다면, 그 패키지는 그 클라이언트에게 주어지지 않으며, 상기 컴퓨터 실행파일(2714)은 삭제 및 덮어쓰기된다(단계 2868).

상기 데이터 파일이 상기 저장소에 흡수된 후, 상기 클라이언트 소프트웨어(2724)는 바람직하게 상기 장치 드라이버 중 하나를 경유하여 상기 상기 컴퓨터 실행파일(2714)을 삭제 및 덮어쓰기한다.

일 실시예에서, 패키지가 유효하다고 결정된 후, 그러나 데이터 파일이 저장소에 흡수되기 전, 상기 장치 드라이버는 저장소내에 상기 데이터 파일에 대한 어소시에이션(association)을 만들기 위하여 클라이언트에게 질문한다. 상기 어소시에이션은 바람직하게 하나의 파일이며, 보다 바람직하게는 실제로 제로-바이트(zero-byte) 길이의 파일인 " 태그" 파일이다. 클라이언트는 통상의 방법으로 그 파일의 이름을 붙일 수 있다. 클라이언트에게, 그 파일은 저장소 내 실제의 데이터 파일을 주는 것처럼 보이지만 그것은 아니다. 만약 그 클라이언트가 상기 태그 파일의 특성에 접근한다면, 다이알로그박스(dialog box; 대화상자)는 상기 데이터 파일에 연계된 하나 또는 그 이상의 승인을 표시한다.

일단 파일이름이 선택되고 그 클라이언트가 그 데이터를 판독하고 저장소로 전송하면, 그 데이터는 상기 데이터 파일을 열어 접근될 수 있다. 이것은 어떠한 사용자의 프로세스에 의하여도 수행될 수 있다. 만약 사용자가 그 파일을 더블-클릭하면, 그러한 파일형태와 연계된 애플리케이션이 자동적으로 시작되고 호출 프로세스를 경유하여 그 파일을 열고자 할 것이다. 상기 클라이언트 소프트웨어(2724)는 상기 호출 프로세스를 차단하고 상기 호출 프로세스 상에서 보안검사를 실행한다. 상기 보안검사는 호출 프로세스가 데이터 파일 내에서 데이터가 누출되게 하는 데이터 홀을 만들지 않았다는 것을 확인한다. " 누설 데이터(leaking data)" 는 보안이 요구되는 데이터를 시스템의 외부로 전달하는 것을 말한다. 데이터 보안이 매우 중요한 애플리케이션에 대하여는 데이터 누출을 제한할 필요가 있다.

상기 호출 프로세스가 보안 체크를 통과하면, 데이터의 필드에 대한 클라이언트의 요구를 확인하기 위하여 다이알로그가 클라이언트에게 표시된다. 하나 또는 그 이상의 승인으로 이루어지는 상기 승인 세트가 표시되며 클라이언트가 동의할 어떤 경고문이 제공된다. 경고문은 예를 들어, 데이터의 필드가 접근되면 저장되지 않은 모든 데이터는 잃게 된다 등의 문구를 포함한다. 클라이언트가 동의하면, 도 27에 도시된 바와 같이, 클라이언트 컴퓨터 시스템(2617)의 환경은 철저하게 변화한다. 상기 시스템(2617)을 작동하는 어떤 프로세스도 상기 시스템(2617) 상에서 수정될 수 없다. 이러한 제한은 데이터 파일에 접근하는 프로세스가 멈추거나 예를 들어, 만료 승인에 의하여 종료될 때 까지 그대로 유지된다.

또 다른 보호 매카니즘은 이하에서 설명되는 스푸핑(spoofing)으로 진행된다.

데이터 보안은 컴퓨터 사용자 및 지적 재산권자에게 심각한 문제이다. 데이터 파일의 보안, 손실 및 권한없는 활동으로부터 데이터의 보호를 위한 암호화와 같은 수단을 사용하는 것이 보통이다.

암호화된 파일과 같이 보안된 파일이 같은 파일 시스템에서 비보안 파일과 나란히 저장되는 것이 알려져 있다. 상기 암호화된 파일은 이름과 크기 등 적당한 파일 속성을 가지고 다른 파일처럼 파일 디렉토리에 있는 것으로 보인다. 그러나, 그 파일에 포함된 데이터는 복호화될 때까지 사용자 애플리케이션이 이해할 수 없다. 더욱이 상기 암호화 프로세스에서 파일의 크기는 암호화되지 않은 본래의 데이터에 비하여 크거나 또는 작게 된다. 이 경우, 파일 크기를 결정하는 파일 시스템에 요구하여 본래 데이터의 실제 크기로 확실하게 복귀시킬 수는 없다. 사용자의 관점에서, 이러한 형태의 데이터 보안은 바람직한 투명성(transparency)이 부족하다.

또한 보안된 파일은 통상의 파일 시스템으로부터 분리된 별도의 물리적 또는 가상의 장소에 저장되는 것이 알려져 있다. 이러한 장소에는 원격 네트워크 장치, 암호화되거나 또는 암호로 보호되는 파일 시스템, 또는 기타 가상의 보안 파일시스템이 포함된다. 이러한 형태의 데이터 보안에서는 보안된 파일과 비보안 파일이 비록 논리적으로는 서로 관련이 있다고 하더라도 사용자가 단일의 파일 디렉토리에서 그 파일들을 자유롭게 섞어 쓸 수 없다. 비록 사용자가 비보호 디렉토리에서 다른 장소에 있는 보안된 파일에 이르는 상징적 링크 또는 최단경로를 설정할 수 있다고 하더라도, 권한이 있는 자의 실행에 혼동을 주는 비합리적인 단계와 보호 데이터에 편리하게 접근하기 위한 노력이 부가된다.

여기에서 사용된 용어 " 디바이스 드라이버" 또는 " 드라이버" 는 하드웨어 장치에 직접적으로 또는 간접적으로 접근하거나 하드웨어를 제어하기 위한 컴퓨터 기반의 명령어(computer-implemented instructions)들을 포함하는 것으로 이해되어야 하며, 상기 명령어는 제한은 없으나 장치 드라이버, 가상의 장치 드라이버(VxDs), NT 커넬 모드 아키텍쳐(kernel mode architecture)에 사용되는 명령어, WIN32 드라이버 모델(WDM)에서 사용되는 명령어, 및 컴퓨터 언어 중에서 컴퓨터, 컴퓨터 아키텍쳐, 네트워크 또는 운영체계를 지지하는 기타의 명령어를 포함한다.

여기에서 사용된 용어 " 정보" 및 " 데이터" 는 각각 매우 넓은 의미를 포함하며, 각각은 텍스트, 오디오 및 비디오 데이터를 포함한다. 구체적인 예를 들면, 용어 " 정보" 는 미가공 데이터, 가공된 데이터 및 미가공 데이터와 가공된 데이터의 조합을 의미할 수 있다.

도면들에서 예시된 실시예가 예시의 목적으로 " 파일시스템 감시자" 로 기재된 장치 드라이버를 포함한다고 하더라도, 여기에서 사용된 용어 " 파일시스템 감시자" 는 본 발명의 파일 스푸핑(spoofing)을 사용하는 종류의 디바이스 드라이버를 통칭한다. 본 발명의 범위 내에서 디바이스 드라이버는 제한은 없으나 일반적인 목적 감시, 승인감시, 필터링, 암호화, 복호화, 바이러스 검출, 데이터 감시, 디바이스로 지시되는 입출력 기능 및 기타 기능을 포함하는 일종의 유용한 기능을 수행하며, 감시 또는 파일 시스템에 관련된 기능에 한정되지 않는다. 파일 스푸핑을 수행하는 디바이스 드라이버는 본 발명의 범위 내에서 적절하게 부과된다.

본 발명의 일 실시예는 윈도우즈 9x 운용체계에 구비된다. 도 29를 참조하면, 윈도우즈 9x 운용체계의 구성요소는 서로 다른 시스템 보호의 레벨을 제공하는 사용자 모드 코드(2800)와 커널 모드 코드(2830)로 구분된다. 일 실시예에 있어서, 상기 사용자 모드 코드(2800)는 16-비트 및 32-비트 소프트웨어 애플리케이션(2821-2822)을 운용할 수 있는 시스템 가상 장치(2820)와 복수의 MS-DOS 가상 장치(2825)를 포함한다. 이 실시예에서, 상기 커널 모드 코드(2830)는 저 레벨 운용체계 서비스와 가상 장치 매니저(vertual machine manager, 2840)와 같은 가상 디바이스 드라이버, 본 발명의 파일시스템 감시자(2850), 및 설치 가능한 파일시스템 관리자(2860)로 이루어진다.

설치 가능한 파일시스템 관리자(2860) 하에 FAT 및 NTFS와 같은 복수의 파일 시스템 드라이버(2870-2872)가 있다. 상기 파일 시스템 드라이버(2870-2872) 하에 블록 입출력 서브시스템(2880)이 있다. 상기 블록 입출력 서브시스템(2880)은 요청들이 상기 파일 시스템 조직, 포트 입출력에 대한 단일화된 드라이버(2882), 및 층을 이룬 복수의 디바이스 드라이버(2883-2884)를 통과하도록 관리하는 입출력 슈퍼바이저(2881)를 포함한다.

이 실시예에서 첫 번째 디바이스 드라이버(2850)는 입출력 요청이 상기 설치 가능한 파일 시스템 매니저(2680)로 보내지기 전에, 사용자 모드 코드(2810) 및 사용자 모드 코드(2810)에서 운용하는 애플리케이션(2821-2822)으로부터의 모든 입출력 요청을 차단한다. 상기 첫 번째 디바이스 드라이버(2850)는 상기 설치 가능한 파일시스템 관리자(2860), 파일시스템 드라이버(2870-2872), 블록 입출력 서브시스템(2880)에서 발생하는 모든 파일시스템 활동을 감시할 수 있으며, 필요하다면 걸러낼 수도 있다. 시스템 초기설정 동안 IFMGR_InastallFileSystemApiHook로의 호출에 의하여, 상기 첫 번째 디바이스 드라이버(2850)는 상기 운영체계가 시작 또는 재시작될 때 그 호출에 연결되며 그 때 그것은 모든 파일시스템 요청의 스택(stack)상에서 기능적으로 최우선의 위치에 삽입된다. 설치 가능한 파일시스템 관리자(2860)로부터 층을 이룬 복수(2883-2884) 내 각 드라이버를 통하여 떨어지는 입출력 요청은 가장 높은 레벨에서 가장 낮은 레벨로 통과되며, 백업(back up) 스택을 입출력 요청의 소스(source)로 통과시킴에 따라 상기 디바이스들은 요청의 결과를 관찰할 수 있다. 상기 스택상의 각 디바이스 드라이버는 입출력 요청을 점검하거나, 또는 입출력 요청을 저 레벨로 통과시키지 않거나, 또는 필요하다면 스스로 새로운 입출력 요청을 생성할 수도 있다. 그러한 디바이스 드라이버는 인터럽트(interrupt) 또는 디바이스가 사용 가능하게 될 때까지 기다리기를 요구하는 기능을 수행한다. 기다리는 동안 상기 디바이스 드라이버는 간단하게 자신의 호출자(caller)로 돌아가 호출 애플리케이션 또는 디바이스 드라이버가 상기 I/O 요청과 병행하여 다른 작업을 수행하도록 한다. 선택적으로, 상기 호출 애플리케이션 또는 디바이스 드라이버는 상기 입출력 요청이 완성될 때까지 단순히 기다릴 수 있다(" 블록").

도 30과 관련하여 예시된 다른 실시예에서, 본 발명은 윈도우즈 NT 운영체계상에 구비될 수 있다. 그 기술분야에서 잘 알려진 바와 같이, 윈도우즈 NT 하에 사용자 모드에서 운용되는 애플리케이션(2900)은 입출력 요청을 운영체계 서비스(2910)에 보낸다. 입출력 관리자(2920)는 입출력 요청을 받고 여러 드라이버간의 입출력 패킷의 전송을 조절한다. 선택적으로, 여러 드라이버는 입출력 관리자(2920) 또는 여러 드라이버간의 정보의 전송을 조절하는 다른 디바이스 없이도 서로간에 직접적으로 통신할 수 있다.

윈도우즈 NT와 같은 운영체계의 통상적인 입출력 시스템은 입출력 요청을 처리하는 복수의 디바이스 드라이버(2930-2932)를 포함한다. 그러한 디바이스 드라이버로 파일시스템 드라이버(2930) 및 층이 형성된 복수의 디바이스 드라이버(2931-2932)를 그 예로 들 수 있다. 상기 입출력 관리자(2920)는 입출력 요청 패킷을 상기 파일시스템 드라이버(2930)에 전달하여 입출력 요청을 관리하도록 한다. 그러나, 그 기술분야에서 잘 알려진 바와 같이, 파일시스템 관리자(2850)는 객체지향 방식에서 다른 디바이스 드라이버(2930-2932)에 첨부될 수 있다. 그에 따라 상기 I/O 매니저는 타겟 디바이스 드라이버(2930-2932)를 향하도록 의도된 입출력 요청 패킷을 상기 타겟 디바이스 드라이버(2930-2932)에 첨부된 파일시스템 관리자(2950)로 발송한다. 이러한 예시적 실시예에서, 상기 파일시스템 관리자(2950)는 복수의 파일시스템 드라이버 객체(2930)에 각각 첨부된다.

도 31은 스푸핑을 사용하는 파일시스템 관리자(2950)에서 데이터 보안을 제공하는 방법의 일 실시예의 플로차트이다.

도 31에 도시된 바와 같이, 단계 3000에서, 본 발명의 파일 스푸핑 프로세스는 파일시스템 요청이 감지될 때마다 개시된다. 단계 3010에서, 상기 프로세스는 상기 파일시스템 요청이 스푸프 파일(spoofed file)을 포함하는 지를 결정한다. 단계 3010의 결정은 요청의 객체로서 지정된 이름이 붙여진 파일(a named file)을 구체적으로 기입한 어떤 파일시스템 요청에 대하여 수행된다. 그러한 파일시스템 요청들은 파일을 열기 위한 FILE_OPEN, 파일을 삭제하기 위한 FILE_DELETE, 및 파일에 새 이름을 붙이기 위한 FILE_RENAME을 포함한다. 윈도우즈 NT 운용체계에서, 그러한 파일

시스템 요청은 또한 파일 정보를 질문하기 위한 FILE_QUERY_INFORMATION 및 파일 정보를 설정하기 위한 FILE_SET_INFORMATION을 포함한다. 이러한 각각의 호출들(calls)은 파일 이름이 구체적일 것을 요구한다. 단계 3010의 결정에서 상기 구체적인 파일 이름은 그 파일이 스푸프 파일인지를 결정하기 위하여 체크된다.

스푸프 파일은 데이터를 사용자에 의하여 쉽게 접근될 수 없는 보안된 파일 장소, 예컨데 보증된 또는 암호화된 가상의 파일 시스템에 위치시킴으로써 보호되어 있는 한편, 상기 파일시스템 중 사용자 접근이 가능한 부분에서 태그파일(tag file)이 플레이스홀더(placeholder)로서 작용하는 것을 유지시키는 파일이다. 디스크 공간을 절약하기 위하여, 상기 태그파일은 길이가 제로 바이트(zero bytes)이다. 일 실시예에서, 사용자의 관점에서 볼 때, 상기 태그파일은 분명히 보안된 데이터를 포함하는 것처럼 보이며, 상기 파일 스푸핑 프로세스는 파일시스템 요청이 상기 보안된 파일 장소에 도달될 수 있도록 한다.

어떤 파일이 스푸프 파일인지를 결정하기 위하여, 상기 프로세스는 모든 스푸프 파일의 데이터베이스에 대하여 그 파일 이름을 검사하고, 만약 그 파일이 스푸프 파일이라면, 상기 프로세스는 상기 태그 파일과 연계된 보안된 파일(the secured file)을 결정한다. 선택적 실시예에서, 상기 프로세스는 태그파일에 저장된 데이터에 의존한다.

만약 단계 3010의 결정이 상기 파일 요청이 스푸프 파일을 포함하는 것이라면 상기 프로세스는 단계 3011에서 계속되고 상기 파일시스템 요청을 수행한다. 일 실시예에서, 상기 프로세스는 다음의 보다 낮은 드라이버를 호출하여 상기 보안된 파일에 대한 파일시스템 요청을 실행한다.

상기 프로세스는 단계 3012에서 계속되어 상기 파일시스템 요청들에 의해 되돌려진 정보를 수정한다. 파일 크기와 같은 파일 특성들은 파일 시스템 요청들에 의해 되돌려진 정보의 부분이다. 상기 프로세스는 되돌려진 정보로부터 태그파일의 선택된 정보특성들을 제거하고, 보안된 파일의 응답 파일 특성을 대신 사용한다. 예를 들어, 태그파일의 파일 크기가 제로인 경우에, 사용자는 그 대신에 상기 응답 보안된 파일이 크기를 볼 것이다. 상기 프로세스는 단계 3040에서 끝난다.

단계 3010으로 돌아와, 만약 단계 3010의 결정이 파일 요청이 스푸프 파일을 포함하지 않는 것이라면, 상기 프로세스는 단계 3020에서 계속되어 상기 파일 시스템 요청이 스푸프 파일을 되돌리는 디렉토리 요청을 포함하는 지를 결정한다. 윈도우즈 9x에서 그러한 파일시스템 요청들은 첫 번째 매칭 파일을 찾기 위한 FIND_OPEN 또는 FILE_FIRST와, 다음 매칭 파일을 찾기 위한 FIND_NEXT를 포함한다. 윈도우즈 NT 운영체계에서는, 그러한 파일시스템 요청은 매칭 파일 이름들의 버퍼(buffer)를 제공하는 DIRECTORY_CONTROL을 포함한다.

만약 단계 3020이 결정이 상기 파일 시스템 요청이 스푸프 파일을 되돌리는 디렉토리 요청을 포함하는 것이라면, 상기 프로세스는 단계 3021에서 계속되어, 파일 시스템 요청을 수행한다. 일 실시예에서, 상기 프로세스는 다음의 보다 낮은 드라이버를 호출하여 태그파일에 대한 파일 시스템 요청을 실행하도록 한다. 선택적인 실시예에서, 상기 프로세스는 파일 시스템 요청을 태그파일 대신에 보안된 파일에 관련한 것으로 고쳐 쓰고, 다음의 보다 낮은 드라이버를 호출하여 보안된 파일에 대한 파일시스템 요청을 실행하도록 한다.

상기 프로세스는 단계 3022에서 계속되어, 파일시스템 요청에 의하여 되돌려진 정보가 어떤 스푸프 파일들에 관련하는지를 결정한다. 첫 번째 매칭 파일 또는 다음의 매칭 파일을 찾기 위한 요청들은 각각 단일의 파일을 되돌릴 것이다. 윈도우즈 NT 파일 시스템에 디렉토리 제어를 위한 요청은 각각 고려되어야 하는 파일 이름들의 한 버퍼를 되돌릴 것이다.

만약 되돌려지지 않는 스푸프 파일이 없다면, 상기 프로세스는 단계 3040에서 종결된다.

만약 어떤 스푸프 파일이 되돌려 진다면, 상기 프로세스는 파일 시스템 요청에 이하여 되돌려진 정보를 수정하여 단계 3025에서 계속된다. 파일 크기와 같은 파일 특성들은 파일시스템 요청에 의하여 되돌려진 정보의 부분이다. 상기 프로세스는 되돌려진 정보으로부터 태그 파일의 선택된 파일 특성을 제거한다. 예를 들어, 태그 파일의 파일크기가 제로인 경우, 사용자는 그 대신에 통신 보안 파일의 파일 크기를 볼 것이다. 그러면 상기 프로세스는 단계 3040에서 종결된다.

단계 3020으로 돌아가, 만약 단계 3020의 결정이 상기 파일 시스템 요청이 스푸프 파일에 되돌려지는 디렉토리 호출을 포함하지 않는 것이라면, 상기 프로세스는 단계 3030에서 계속되어 파일 시스템 요청을 수행하고, 단계 3040에서 종결된다.

이와 같이, 본 발명은 다양한 변형과 조합이 가능하고, 본 명세서에서 보인 특별한 실시예에 제한되지 않는 것임을 알 수 있다. 또한, 여러 실시예에 개시된 각각의 세그먼트들은 모두가 단일 실시예에서 제공될 필요는 없으며, 오히려 필요한 경우에 필요한 세그먼트의 조합으로 제공될 수 있다. 본 발명에 따른 시스템은 전적으로 또는 부분적으로 특별한 목적의 하드웨어 또는 종래의 일반적인 목적의 하드웨어 또는 그들의 조합, 적절한 프로그램으로 제어될 수 있는 부분으로 구축될 수 있다. 프로그램은 전적으로 또는 부분적으로 통상의 방법으로 시스템 상에 저장되거나, 또는 전적으로 또는 부분적으로 네트워크 또는 통상의 방법으로 정보를 전달하기 위한 다른 메카니즘상에서 시스템으로 제공된다. 따라서, 본 발명의 상술한 설명은 해당 기술분야의 기술자에 의하여 충분히 수정, 변경 및 적용될 수 있으며, 그러한 수정, 변경 및 적용들은 본 발명의 범위 내로 인정되어야 할 것이며, 그것은 첨부된 청구항에 의하여 공표될 것이다. 상술된 본 발명에 따라, 본 발명의 정신과 범위에서 벗어나지 않는 많은 방법으로 변경될 수 있다는 것은 해당 기술 분야의 기술자에게 자명할 것이다. 그러한 수정들은 다음의 청구항의 범위에 속한다고 할 것이다.

도면의 간단한 설명

도 1. 데이터보안을 위한 시스템과 방법의 일 실시예를 나타낸 블록다이어그램.

도 2. 본 발명의 일 실시예에 따른 보호되는 데이터전송시스템부의 블록다이어그램.

도 3. 본 발명의 일 실시예에 따른 응용프로그램 인터페이스 콤포넌트와 디바이스 드라이버 사이의 상호작용을 나타낸 블록다이어그램.

도 4A-C. 본 발명의 일 실시예에 따른 레지스터리 모니터링 시스템의 플로우차트.

도 5A-D. 본 발명의 일 실시예에 따른 할당된 메모리 블록킹 방법의 플로우차트.

도 6. 본 발명의 백-채널링 방법의 일 실시예에 따른 컴퓨터시스템 작동의 블록다이어그램.

도 7. 본 발명의 백-채널링 방법의 일 실시예에 따른 메모리저장시스템에서 파일 오픈의 플로우차트.

도 8. 본 발명의 백-채널링 방법의 일 실시예에 따른 메모리저장시스템에서 파일 읽기/쓰기의 플로우차트.

도 9. 본 발명의 백-채널링 방법의 일 실시예에 따른 메모리저장시스템에서 파일정보요구의 플로우차트.

도 10. 본 발명의 백-채널링 방법의 일 실시예에 다른 메모리저장시스템에서 파일변환요구의 플로우차트.

도 11. 윈도우즈 9x 운영체제의 시스템 구조 레이아웃의 블록다이어그램.

도 12. 윈도우즈 NT 운영체제의 시스템 구조 레이아웃의 블록다이어그램.

도 13. 본 발명의 스택의 위치를 정하는 방법의 일 실시예를 설명하는 플로우차트.

도 14. 본 발명의 스택의 위치를 정하는 방법의 리-후킹 단계를 설명하는 플로우차트.

도 15. 본 발명의 일 실시예에 따른 시스템 클럭의 변화 또는 변경을 모니터링하기위한 방법의 플로우다이어그램.

도 16. 선결된 트랙킹 간격에서 시스템 클럭값을 점검하므로서 시스템 클럭의 변화 또는 변경을 모니터링하기위한 방법의 플로우다이어그램.

도 17. 본 발명의 포트 블록킹 방법의 일 실시예에 따른 컴퓨터 시스템 작동의 개요도.

도 18. 본 발명의 포트 블록킹 방법의 일 실시예에 따른 컴퓨터 시스템 작동에서 포트 요청의 플로우차트.

도 19(a). 본 발명의 포트 블록킹 방법의 일 실시예에 따른 컴퓨터 시스템 작동에서 포트 열기 요청의 플로우차트.

도 19(b). 본 발명의 포트 블록킹 방법의 일 실시예에 따른 컴퓨터 시스템 작동에서 포트 닫기 요청의 플로우차트.

도 19(c). 본 발명의 포트 블록킹 방법의 일 실시예에 따른 컴퓨터 시스템 작동에서 보안검사의 플로우차트.

도 20. 본 발명의 네트워크 블록킹 방법의 일 실시예에 따른 컴퓨터 시스템 작동의 개요도.

도 21. 본 발명의 네트워크 블록킹 방법의 일 실시예에 따른 컴퓨터 시스템 작동에서 네트워크 요청의 플로우차트.

도 22. 본 발명의 원리에 따른 파일시스템으로부터 calls를 인터셉트하기위해 삽입되는 은폐 드라이버의 다이어그램.

도 23. 호출 프로세싱이 파일시스템 관리자로부터 체인 아래로 이동하는 선택된 형태의 호출상에서 실행되고 파일시스템 관리자에서 체인 위로 이동하여 리턴되는 본 발명의 일 실시예를 나타낸 플로우차트.

도 24. 윈도우즈 95 운영체제의 파일시스템 논리계층의 다이어그램.

도 25. 윈도우즈 95 운영체제의 파일시스템 논리계층들 사이의 일반적인 파일시스템 요청 체인의 다이어그램.

도 26A-26G. 도 2의 리턴 처리순서와 프로세싱 콜 5의 플로우차트.

도 27. 본 발명의 일 실시예에 따른 정보의 패키지를 전달하는 시스템의 플로우다이어그램.

도 28A-28B. 본 발명의 또 다른 실시예에 따른 전자 패키지를 전달하기위한 방법의 플로우다이어그램.

도 29. 윈도우즈 9x 운영체제의 시스템 구조 레이아웃의 다이어그램.

도 30. 윈도우즈 NT 운영체제의 시스템 구조 레이아웃의 다이어그램.

도 31. 본 발명의 파일 스푸핑 방법의 일 실시예를 나타낸 플로우차트.

(57) 청구의 범위

청구항 1.

데이터 사용을 통제하기 위하여 하나 이상의 승인과 데이터로 이루어진 패키지를 생성하는 단계와;

상기 패키지를 처리하고 저장소에 상기 데이터를 저장하기 위하여 리시버를 제공하는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 2.

제 1 항에 있어서, 상기 패키지를 처리하는 단계가 상기 패키지 열기와 상기 패키지의 처리에 대한 리시버 확인으로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 3.

제 2 항에 있어서, 상기 패키지 읽기에 대하여 적어도 하나의 드라이브 찾기를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 4.

제 1 항에 있어서, 상기 하나 이상의 승인의 위반을 검사하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 5.

제 4 항에 있어서, 리시버 제공에 대한 단계는 내부보안을 제공하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 6.

제 5 항에 있어서, 상기 내부보안이 상기 데이터에 일치하는 태그파일을 생성하는 단계와, 상기 데이터의 실제 파일이름과 상기 태그파일에 대해 일치하는 태그이름을 포함하는 가상 테이블에서 상기 데이터에 태그파일을 맵핑하는 단계로 이루어지며,

상기 가상 테이블과 상기 데이터는 상기 저장소에 저장되는 것을 특징으로 하는 데이터보안 유지방법.

청구항 7.

제 5 항에 있어서, 내부보안을 제공하는 단계는

적어도 하나의 상기 저장소, 상기 패키지의 읽기를 위해 사용된 드라이버, 상기 승인을 저장하는 데이터베이스에 대해서 초기 위치와 일치하는 고정 어드레스를 확인하는 단계와;

시스템 작동을 통제하기 위해서 키를 제공하는 것과 함께 상기 어드레스를 결합시키는 단계;

키가 작동되지 않을 때 확인하는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 8.

제 5 항에 있어서, 패키지 생성의 단계는 상기 패키지가 열기될 때 상기 리시버의 작동 확인에 대한 실행을 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 9.

제 5 항에 있어서, 상기 내부보안은

호출 프로세스에서 레지스트리 키에 대한 처리를 요청하는 단계;

상기 처리에 대한 레지스트리 키값을 요청하는 단계;

상기 요청을 완료시키기 위한 보안해제 획득하는 단계로 이루어진 레지스트리에 대한 감시단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 10.

제 9 항에 있어서, 호출 프로세스에서 레지스트리 키에 대한 핸들을 요청한 후,

프로세스 아이디와 레지스트리 키를 결정하는 단계;

상기 프로세스가 보호 프로세스 리스트를 검사하므로서 보호되는지를 결정하는 단계;

상기 프로세스가 보호된다면, 상기 레지스트리 키가 거절 리스트에 있는지를 결정하는 단계;

상기 레지스트리 키가 거절 리스트에 있다면, 상기 프로세스가 상기 레지스트리 키를 엑세스하는 것을 거부하는 단계;

상기 프로세스가 상기 보호 리스트상에 없거나 상기 레지스트리 키이름이 거절 리스트상에 없다면, 상기 요청을 완료하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 11.

제 9 항에 있어서, 상기 핸들에 대한 레지스트리 키값을 요청한 후,

프로세스 아이디와 레지스트리 키값을 결정하는 단계;

상기 프로세스가 상기 보호 프로세스 리스트를 검사하므로서 보호되는지를 결정하는 단계;

상기 프로세스가 보호된다면, 상기 레지스트리 키가 거절 리스트상에 있는지를 결정하는 단계;

상기 레지스트리 키가 거절 리스트상에 있다면, 상기 프로세스가 상기 레지스트리 키값을 엑세스하는 것을 거부하는 단계;

상기 프로세스가 상기 보호 리스트상에 없다면, 상기 요청을 완료하는 단계;

상기 레지스트리 키가 상기 거절 리스트상에 없고 상기 프로세스가 상기 보호 프로세스 리스트상에 있다면, 상기 값 요청을 처리하고 상기 값이 거절 리스트상에 있는지를 결정하는 단계;

상기 값이 거절 리스트상에 없다면 상기 요청이 완료되도록 허가하는 단계;

상기 값이 거절 리스트상에 있다면 상기 레지스트리 키값에 대한 엑세스를 거부하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 12.

제 9 항에 있어서, 핸들과 값이 변경되고 삭제된 후,

프로세스 아이디를 결정하는 단계;

상기 프로세스가 상기 보호 프로세스 리스트상에 있는지를 검사하므로서 상기 프로세스가 보호되는지를 결정하는 단계;

상기 프로세스가 상기 보호 프로세스 리스트상에 없다면, 상기 요청이 완료되는 단계;

상기 프로세스가 상기 보호 프로세스 리스트상에 있다면, 상기 요청이 완료되도록 허가하지 않는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 13.

제 5 항에 있어서, 내부보안을 제공하는 단계가

요청 프로세스에 대하여 메모리 페이지를 확보하기 위한 호출을 제공하는 단계;

상기 페이지가 공유될 수 있는지에 따라 상기 예비호출를 필터링하는 단계;

상기 요청 프로세스 또는 다음의 요청 프로세스에 대하여 상기 메모리 페이지를 위임하는 호출을 제공하는 단계;

상기 페이지가 공유될 수 있는지 그리고 상기 프로세스가 보호될 수 있는지에 따라 상기 위임 호출을 필터링하는 단계로 이루어진 공유메모리를 감시하는 방법으로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 14.

제 13 항에 있어서, 상기 예비호출을 필터링하는 단계는

상기 페이지가 요청 매개변수를 기반으로 공유될 수 있는지를 결정하는 단계;

상기 페이지가 공유될 수 없다면, 상기 요청이 완료되도록 허가하는 단계;

상기 페이지가 공유될 수 있다면, 레코드 생성에 의해서 상기 예비호출이 트랙킹하고 상기 레코드를 공유메모리 리스트에 입력시키는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 15.

제 14 항에 있어서, 상기 레코드는 프로세스 아이디, 페이지 번호와 공유 카운트를 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 16.

제 13 항에 있어서, 상기 위임 호출을 필터링하는 단계는

공유메모리 리스트를 엑세스하므로서, 상기 페이지가 다른 프로세스에 의해 공유되는지를 결정하는 단계;

상기 페이지가 공유된다면, 보호 프로세스 리스트를 엑세스하므로서 상기 공유 프로세스의 둘 중 하나가 보호되는지를 결정하는 단계;

둘 중 하나의 프로세스가 보호된다면, 페이지가 공유되는 것을 허가하지 않는 단계;

양쪽 모두의 프로세스가 보호되지 않는다면, 새 공유메모리 레코드를 생성하고,

상기 페이지를 공유하고 상기 새 레코드에 포함된 정보와 함께 상기 공유메모리 리스트를 업데이트하기 위한 공유 카운트를 업데이트하는 단계;

상기 페이지가 공유되지 않는다면, 상기 위임 요청을 완료시키는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 17.

제 16 항에 있어서, 상기 레코드는 프로세스 아이디와, 페이지 번호, 공유 카운트를 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 18.

제 13 항에 있어서,

모든 어드레스 공간의 상기 메모리 페이지를 자유롭게하는 호출을 제공하는 단계;

보호 프로세스 리스트를 검사하므로서 상기 프로세스가 보호되는지를 결정하는 단계;

상기 프로세스가 보호된다면, 보호 데이터를 삭제하기 위하여 상기 페이지를 덮어쓰기하고,

상기 덮어쓰기된 페이지와 동일한 페이지 번호와 함께 상기 공유메모리 리스트에서 모든 레코드를 삭제하는 단계;

상기 프로세스가 보호되지 않는다면, 비보포 프로세스 페이지와 일치하는 페이지 번호와 함께 공유메모리 리스트로부터 모든 레코드를 삭제하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 19.

제 5 항에 있어서, 내부보안을 제공하는 단계는

다른 시스템 데이터로부터 저장 데이터를 분리시키는데 대하여 저장시스템을 제공하는 저장제공단계;

차단 파일시스템 호출 중 각각의 특정한 것에 대하여 상기 차단 파일시스템 호출의 상기 특정한 것이 상기 저장 데이터를 엑세스하는 프로세스로부터인지를 결정하고, 상기 차단 파일시스템 호출의 상기 특정한 것이 상기 저장 데이터를 엑세스하는 프로세스로부터이다면 상기 파일시스템이 상기 저장 시스템내에서만 데이터를 생성하거나 변경하도록 허가하는 파일시스템 호출을 차단하는 파일시스템 보안드라이버를 제공하는 파일시스템 보안드라이버 제공단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 20.

제 19 항에 있어서, 상기 파일시스템 보안드라이버.제공단계는 파일 열기 핸들링 단계를 더 포함하며,

파일열기 호출인 상기 차단 파일시스템 호출의 어떤 특정한 하나에 대하여상기 파일 열기 핸들링 단계는,

상기 파일 열기 호출이 상기 저장 데이터로부터 데이터에 대한 요청인지를 결정하는 단계;

상기 파일 열기 호출이 상기 저장 데이터로부터 데이터에 대한 요청이라면 상기 프로세스가 상기 저장 데이터로부터 상

기 데이터를 이미 열었던 보호 프로세스인지를 확인하기 위하여 상기 요청을 만드는 프로세스상에서 검사를 수행하고, 만약 그렇다면 상기 저장 데이터에 대한 엑세스를 허가하고 상기 요청을 만든 프로세스상에서 엑세스 검사를 실행시키며, 그때 상기 엑세스 검사가 통과된다면 이미 보호 프로세스가 아닌 상기 프로세스에 대한 엑세스를 허가하므로서 상기 요청을 처리하나, 상기 엑세스 검사가 통과되지 않는다면 엑세스를 전혀 허가하지 않는 단계;

상기 파일 열기 호출이 상기 저장 데이터로부터 데이터에 대한 요청이 아니라면 상기 프로세스가 이미 보호 프로세스인지를 확인하기 위하여 상기 요청을 만드는 상기 프로세스상에서 검사를 실행하고, 상기 요청을 만든 상기 프로세스가 보호 프로세스가 아니라면 상기 요청을 운영체제로 통과시키며, 상기 요청을 만든 상기 프로세스가 보호 프로세스라면 상기 파일 열기 호출에서 언급된 파일이 존재하는지를 결정하고, 존재한다면 읽기에서만 상기 파일을 열고, 존재하지 않는다면 상기 저장 데이터에서 상기 파일을 생성시키는 단계;로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 21.

제 20 항에 있어서, 상기 저장 데이터에 대한 엑세스를 이전에 승인되지 않은 상기 프로세스에 대한 엑세스를 허가하므로서 상기 요청에 대한 상기 프로세싱은,

사용자가 상기 저장 데이터로부터 상기 테이터를 열고자하는지를 결정하기 위해 상기 사용자에게 질문하는 단계;

상기 사용자가 상기 데이터를 열고자 한다면 상기 저장 데이터로부터 상기 데이터를 여는 단계를 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 22.

제 21 항에 있어서, 상기 저장 데이터에 대한 엑세스를 이전에 승인되지 않은 상기 프로세스에 대한 엑세스를 허가하므로서 상기 요청에 대한 상기 프로세싱은,

상기 허가된 엑세스를 기록하고 허가된 전체 엑세스를 감시하는 단계를 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 23.

제 20 항에 있어서, 상기 저장 데이터에 대한 엑세스를 이전에 승인되지 않은 상기 프로세스에 대한 엑세스를 허가하므로서 상기 요청에 대한 상기 프로세싱은,

상기 저장 데이터를 엑세스하도록 허가된 프로세스의 리스트에서 상기 요청을 만든 상기 저장 데이터에 대한 엑세스를 이전에 승인될 수 없도록 한 상기 프로세스를 기록하는 단계를 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 24.

제 20 항에 에 있어서, 상기 저장 데이터에 상기 파일을 생성하는 상기 단계는,

상기 보호 프로세스에 우선적인(stand-in) 파일 핸들을 보내는 단계

상응하는 저장 파일 핸들을 생성하는 단계;

상기 우선적인 파일 핸들과 상기 상응하는 저장 파일 핸들을 저장하는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 25.

제 20 항에 있어서, 읽기에 대해서 상기 파일을 열기하는 상기 단계는,

상기 파일의 표시된 변경이 허가된 상기 파일열기 호출의 어떤 파일 요청 플래그를 변경하는 단계;

상기 변경된 파일열기 호출을 운영체제로 통과시키는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 26.

제 19 항에 있어서, 상기 파일시스템 보안드라이버 제공단계는,

파일읽기/파일쓰기 호출인 상기 차단 파일시스템 호출의 어떤 특정한 것에 대하여,

상기 읽기/쓰기 요청이 상기 저장 데이터로부터 데이터에 대한 요청인지를 결정하는 단계;

상기 읽기/쓰기 요청이 상기 저장 데이터로부터 데이터에 대한 요청이라면, 상기 요청을 만드는 프로세스가 상기 저장 데이터를 엑세스하도록 허가될지에 대한 엑세스를 허가하는 단계;

상기 읽기/쓰기 요청이 상기 저장 데이터로부터 데이터에 대한 요청이 아니라면, 상기 요청을 만든 상기 프로세스가 상기 저장 데이터를 엑세스하도록 허가되지 않을지에 대한 엑세스를 허가하는 단계, 그리고 상기 읽기/쓰기 요청이 읽기 요청인지를 엑세스하도록 허가하는 단계로 이루어진 파일읽기/파일쓰기 요청 핸들링 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 27.

제 19 항에 있어서, 상기 파일시스템 보안드라이버 제공단계는

상기 파일정보 요청이 상기 저장 데이터로부터에 고나한 요청인지를 결정하는 단계;

그렇지 않다면, 상기 파일정보 요청을 상기 운영체제로 통과시키는 단계;

그렇다면, 정확한 파일크기를 식별하고 상기 정확한 파일크기를 리턴하는 단계로 이루어진 파일정보 요청단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 28.

제 19 항에 있어서, 상기 파일시스템 보안드라이버 제공단계는

상기 파일변경 요청이 상기 저장 데이터로부터의 데이터에 관한 요청인지를 결정하는 단계;

그렇다면, 상지 저장 데이터에서 상기 파일변경 요청을 수행하는 단계;

그렇지 않다면, 상기 요청 프로세스가 보호 프로세스인지를 확인하기 위해 검사하는 단계;

그렇지 않다면, 상기 파일변경 요청을 상기 운영체제로 통과시키는 단계;

그렇다면, 상기 요청을 차단하는 단계로 이루어진 파일변경 요청단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 29.

제 19 항에 있어서, 상기 파일시스템 보안드라이버 제공단계는,

파일열기 호출인 상기 차단 파일시스템 호출의 어떤 특정한 것에 대하여,

상기 파일열기 호출이 상기 저장 데이터로부터 데이터에 대한 요청인지를 결정하는 단계;

상기 파일열기 호출이 상기 저장 데이터로부터 데이터에 대한 요청이라면,

상기 프로세스가 상기 저장 데이터로부터 이전에 상기 데이터가 열기된 보호 프로세스인지를 확인하기 위하여 상기 요청을 만드는 프로세스에서 검사를 수행하고, 상기 저장 데이터에 대한 엑세스를 허가하고, 상기 요청을 만드는 프로세스에서 엑세스 검사를 수행하며, 그때 상기 엑세스 검사가 통과되면 보호 프로세스가 아니지만 상기 엑세스 검사가 통과되지 않는다면 전혀 엑세스를 허가하지 않도록 하는 상기 프로세스에 대한 엑세스를 허가하므로서 상기 요청을 처리하는 단계;

상기 파일열기 호출이 상기 저장 데이터로부터 데이터에 대한 요청이 아니라면,

상기 프로세스가 이미 보호 프로세스인지를 확인하기 위하여 상기 요청을 만드는 상기 프로세스에서 검사를 수행하고, 상기 요청을 만드는 상기 프로세스가 보호 프로세스가 아니라면 상기 요청을 운영체제로 통과시키고, 상기 요청을 만드는상기 프로세스가 보호 프로세스라면 상기 파일열기 호출에서 언급된 파일이 존재하는지를 결정하며, 상기 파일이 존재한다면 읽기만으로 상기 파일을 연고, 상기 파일이 존재하지 않는다면 상기 저장 데이터에서 상기 파일을 생성하는 단계로 이루어진 파일열기 핸들링 단계:

파일읽기/파일쓰기 호출인 상기 차단 파일시스템 호출의 어떤 특정한 것에 대해서,

상기 읽기/쓰기 요청이 상기 저장 데이터로부터 데이터에 대한 요청인지를 결정하는 단계;

상기 읽기/쓰기 요청이 상기 저장 데이터로부터 데이터에 대한 요청이라면, 상기 요청을 만드는 프로세스가 상기 저장 데이터를 엑세스하도록 허가하는지를 엑세스하는 것을 허가하는 단계;

상기 읽기/쓰기 요청이 상기 저장 데이터로부터 데이터에 대한 요청이 아니라면, 상기 요청을 만드는 상기 프로세스는 상기 저장 데이터를 엑세스하는 것을 허가하지 않는지를 엑세스하도록 허가하는 단계;

상기 읽기/쓰기 요청이 읽기 요청인지를 엑세스하도록 허가하는 단계로 이루어진 파일읽기/파일쓰기 요청 핸들링 단계:

파일정보 요청이 상기 저장 데이터로부터 데이터에 관한 요청인지를 결정하는 단계;

그렇지 않다면, 상기 파일정보 요청을 상기 운영체제로 통과시키는 단계;

그렇다면, 정확한 파일크기를 식별하고 상기 정확한 파일크기를 리턴하는 단계로 이루어진 파일정보 요청단계:

파일변경 요청이 상기 저장 데이터로부터 데이터에 관한 요청인지를 결정하는 단계;

그렇다면 상기 저장 데이터에서 상기 파일변경 요청을 수행하는 단계;

그렇지 않다면 상기 요청 프로세스가 보호 프로세스인지를 확인하기 위하여 검사하는 단계;

그렇지 않다면 상기 파일변경 요청을 상기 운영체제로 통과시키는 단계;

그렇다면, 상기 요청을 차단하는 단계로 이루어진 파일변경 요청단계를 더 포함하여 구성된 것을 특징으로 하는 데이터 보안 유지방법.

청구항 30.

제 5 항에 있어서, 내부보안을 제공하는 단계는

시스템 시계로부터 처음 시간값을 읽는 단계;

하나 이상의 시계-관계의 승인 필드를 가지는 승인 데이터베이스, 하나이상의 시계-관계의 승인을 이루는 각각의 필드와, 저장된 시간값을 이루는 저장된 시간값 필드가 컴퓨터 시스템에서 초기화되는지를 결정하는 단계;

상기 승인 데이터베이스가 초기화된다면 상기 처음 시간값을 상기 저장된 시간값과 비교하고, 상기 처음 시간값이 상기 저장된 시간값보다 더 늦다면 상기 처음 시간값을 상기 저장된 시간값 필드에 저장하고, 상기 처음 시간값이 상기 저장된 시간값보다 빠르다면 상기 하나이상의 시계-관계의 승인은 사용할 수 없게되고, 그것에 따라 상기 시계-관계의 승인이 상기 데이터에 대한 엑세스를 차단하는 기능을 억제하는 단계;

상기 승인 데이터베이스가 초기화되지 않는다면, 상기 처음 시간값을 상기 저장된 시간값 필드에 저장하는 단계로 이루어진 시계 감시자를 초기화하는 단계로 이루어진 데이터에 대한 권한없는 엑세스를 방기하기 위해서 컴퓨터의 상기 시스템 시계를 감시하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 31.

제 30 항에 있어서, 상기 승인 데이터베이스가 초기화되는지를 결정하는 단계는,

상기 승인 데이터베이스에서 상기 저장된 시간값 필드로부터 상기 저장된 시간값을 읽는 단계;

상기 저장된 값이 0(zero)이라면, 상기 승인 데이터베이스가 초기화되지 않도록 종결짓는 단계;

상기 저장된 시간값 필드가 0이 아니라면, 상기 승인 데이터베이스가 초기화되도록 종결짓는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 32.

제 30 항에 있어서,

상기 시계 감시자의 초기화로부터 측정된 내부경과시간상에서 상기 저장된 시간값인 실제 시스템 시간을 트랙킹(tracking)하는 단계;

선결된 트랙킹 간격후에, 상기 시스템 시간으로부터 두번째 시간값을 읽는 단계;

상기 두번째 시간값을 상기 실제 시스템 시간과 비교하고, 상기 비교에 기반된 시간 이탈을 발생시키는 단계;

상기 시간 이탈이 허용할 수 있는 이탈내에 있지 않다면, 상기 하나 이상의 시계-관계의 승인 기능이 억제되는 단계;

상기 시간 이탈이 허용할 수 있는 이탈 내에 있다면, 상기 시계-관계의 승인을 실시하는 단계;

상기 실제 시스템 시간을 저장하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 33.

제 32 항에 있어서, 상기 시간 이탈이 허용할 수 있는 이탈내에 없는지에 대한 단계 이후에, 하나 이상의 시계-관계의 승인 기능을 억제시키며,

상기 시스템 시계로부터 세번째 시간을 읽는 단계;

상기 세번째 시간값을 상기 내부경과시간과 비교하는 단계;

상기 비교를 기반으로한 두번째 시간 이탈을 발생시키는 단계;

상기 두번째 시간 이탈이 상기 허용할 수 있는 이탈내에 있다면, 상기 시계-관계의 승인을 재가능하게 하고, 상기 실제 시스템 시간을 상기 저장된 시간값 필드에 저장하며, 최근의 정확한 시스템 시간값 필트인 세번째 시간값을 상기 승인 데이터베이스에 저장하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 34.

제 32 항에 있어서, 상기 선결된 트랙킹 간격은 대체적으로 0초에서 6초 사이의 범위내에 있는 것을 특징으로 하는 데이터보안 유지방법.

청구항 35.

제 32 항에 있어서, 상기 허용된 이탈은 대체적으로 0초에서 3시간 사이의 범위내에 있는 것을 특징으로 하는 데이터보안 유지방법.

청구항 36.

제 30 항에 있어서, 상기 시계 감시자의 초기화로부터 측정된 실제 시스템 시간상에서 상기 저장된 시간값인 실제 시스템 시간을 트랙킹하는 단계;

상기 시스템 시계로부터 두번째 시간값을 읽는 단계;

두번째 시간값을 상기 실제 시스템 시간과 비교하고, 상기 비교를 기반으로 시간 이탈을 발생시키는 단계;

상기 시간 이탈이 허용할 수 있는 이탈내에 있다면, 상기 두번째 시간값을 상기 저장된 시간값 필드에 저장하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 37.

제 36 항에 있어서, 컴퓨터 전원을 끄는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 38.

제 36 항에 있어서, 상기 허용된 이탈이 대체적으로 0초에서 3시간 사이의 범위내에 있는 것을 특징으로 하는 데이터보안 유지방법.

청구항 39.

제 30 항에 있어서, 상기 시계-관계의 승인이 날짜-관계의 승인으로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 40.

제 5 항에 있어서, 데이터 저장장치에서 데이터를 엑세스하기 위하여 다수의 층을 이룬 디바이스 드라이버를 가지는 컴퓨터 운영체제에 실시가능하도록 설치된 첫번째 디바이스 드라이버에서 데이터보안을 제공하는데 대해서,

상기 첫번째 디바이스 드라이버에서 입출력 요청을 감지하는 단계;

상기 첫번째 디바이스 드라이버가 기능적으로 층을 이룬 다수의 디바이스 드라이버에서 최상위인지를 결정하는 단계;

상기 첫번째 디바이스 드라이버가 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위이라면, 상기 첫번째 디바이스 드라이버에서 상기 입출력 요청을 수행하는 단계;

상기 첫번째 디바이스 드라이버가 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위가 아니라면, 상기 첫번째 디바이스 드라이버에서 상기 입출력 요청을 거부하고, 상기 층을 이룬 다수의 디바이스 드라이버에서 다음 하위-레벨 디바이스 드라이버에 의해 수행되는 상기 입출력 요청을 허가하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 41.

제 40 항에 있어서, 상기 첫번째 디바이스 드라이버는 파일시스템 감시자인 것을 특징으로 하는 데이터보안 유지방법.

청구항 42.

제 40 항에 있어서, 상기 데이터는 보호 가상 파일시스템에 저장되고, 상기 입출력 요청을 실행하는 단계는 데이터보안 측정을 실행하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 43.

제 40 항에 있어서, 상기 데이터는 암호화된 형태로 저장되고, 상기 입출력 요청을 실행하는 단계는 상기 데이터를 복호화하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 44.

제 40 항에 있어서, 상기 입출력 요청을 실행하는 단계는 바이러스에 대한 상기 데이터를 검사하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 45.

제 40 항에 있어서, 상기 첫번째 디바이스 드라이버가 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위에 있는지를 결정하는 단계는

상기 첫번째 디바이스 드라이버가 이전에 호출되었는지를 결정하는 단계;

상기 첫번째 디바이스 드라이버가 이전에 호출되지 않았다면, 초기의 호출 모듈 어드레스를 감지하고, 상기 초기 호출 모듈 어드레스를 저장하며, 상기 첫번째 디바이스 드라이버가 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위인 것으로 결정하는 단계;

상기 첫번째 디바이스 드라이버가 이전에 호출되었다면, 두번째 호출 모듈 어드레스를 감지하고, 상기 두번째 호출 모듈 어드레스를 상기 초기의 호출 모듈 어드레스와 비교하며,

상기 초기 호출 모듈 어드레스가 상기 두번째 호출 모듈 어드레스와 일치한다면 상기 첫번째 디바이스 드라이버가 기능적으로 층을 이룬 다수의 디바이스 드라이버에서 최상위인 것으로 결정하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 46.

제 40 항에 있어서, 상기 보호 첫번째 디바이스 드라이버에서 상기 입출력 요청을 거부하는 단계는,

첫번째 디바이스 드라이버 셧다운 플래그(shutdown flag)를 성정하고, 리-훅(re-hook) 프로세스를 초기화하는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 47.

제 40 항에 있어서, 상기 첫번째 디바이스 드라이버에서 입출력 요청을 감지하는 단계 이후에,

첫번째 디바이스 드라이버 셧다운 플래그가 설정되는지를 검사하는 단계;

상기 첫번째 디바이스 드라이버 셧다운 플래그가 설정된다면, 상기 첫번째 디바이스 드라이버에서 다음 단계를 생략하고, 상기 층을 이룬 다수의 디바이스 드라이버에서 다음 하위-레벨 디바이스 드라이버에 의해 실행되는 상기 입출력 요청을 허가하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 48.

제 47 항에 있어서, 리-훅 프로세스를 초기화하는 단계는

상기 리-훅 프로세스가 초기화된 횟수를 세는 단계;

상기 횟수가 선결된 최고값 문턱에 도달했는지를 검사하는 단계;

상기 횟수가 선결된 최고값 문턱에 도달했다면, 프로그램가능한 보안 응답을 초기화하는 단계;

상기 횟수가 선결된 최고값 문턱에 도착하지 않았다면, 상기 첫번째 디바이스 드라이버의 재연결을 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위로 초기화하는 단계;

상기 첫번째 디바이스 드라이버가 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위에 있다면, 상기 첫번째 디바이스 드라이버 셧다운 플래그를 설정해제하는 단계;

상기 리-훅 프로세스를 종결짓는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 49.

제 48 항에 있어서, 상기 프로그램가능한 보안 응답은 상기 데이터를 손상시키는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 50.

제 48 항에 있어서, 상기 데이터는 보호 가상 파일시스템에 저장되고, 상기 데이터를 손상시키는 단계는 상기 보호 가상 파일시스템을 손상시키는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 51.

제 48 항에 있어서, 상기 프로그램가능한 보안 응답은 열기 애플리케이션을 종결짓는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 52.

제 48 항에 있어서, 상기 프로그램가능한 보안 응답은 상기 데이터 저장장치에서 상기 첫번째 디바이스 드라이버를 손상시키는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 53.

제 48 항에 있어서, 상기 프로그램 가능한 보안 응답은 상기 컴퓨터의 작동을 정지시키는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 54.

제 48 항에 있어서, 상기 프로그램가능한 보호 응답은 상기 컴퓨터가 재부팅을 하도록 하는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 55.

제 1 항에 있어서, 프로세스에 의해 보내지는 포트의 사용에 대하여 포트요청을 감지하는 포트요청 감지단계;

상기 요청 프로세스의 동일성을 결정하는 프로세스 확인단계;

상기 프로세스가 상기 포트를 엑세스하도록 허가되는지를 결정하는 프로세스 검사단계;

상기 프로세스가 상기 포트를 엑세스하도록 허가된다면 상기 포트요청이 실행되도록 허가하고 상기 프로세스가 상기 포트를 엑세스하는 것을 허가하지 않는다면 상기 포트요청을 거부하는 허가/거부 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 56.

제 55 항에 있어서, 상기 프로세스 검사단계는

상기 프로세스가 보호 프로세스 리스트상에 나타나는지를 결정하는 보호 프로세스 리스트 검사단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 57.

제 55 항에 있어서, 상기 포트 요청을 트랙킹(tracking)하는 트랙킹 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 58.

제 5 항에 있어서, 내부보안을 제공하는 단계는,

프로세스에 의해 보내진 포트의 사용에 대하여 포트요청을 감지하는 포트요청 감지단계;

상기 포트요청이 열기 포트요청이라면, 상기 요청 프로세스의 동일성을 결정하는 열기 포트 프로세스 확인단계;

상기 포트요청이 열기 포트요청이라면, 상기 프로세스가 상기 포트를 열도록 허가되는지를 결정하는 열기 포트 프로세스 검사단계;

상기 포트요청이 열기 포트요청이라면 상기 열기 포트요청이 실행되도록 허가하고, 상기 프로세스가 상기 포트를 열도록 허락한다면 상기 열기 포트 요청을 트랙킹하며, 상기 프로세스가 상기 포트를 열도록 허가하지 않는다면 상기 포트요청을 거부하는 열기 포트 허가/거부 단계;

상기 포트요청이 닫기 포트요청이라면, 상기 포트요청을 완료하는 닫기 포트 프로세스 완료단계;

상기 포트의 닫기를 기록하는 닫기 포트 기록단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 59.

제 58 항에 있어서, 상기 열기 포트 프로세스 검사단계는

상기 프로세스가 보호 프로세스 리스트상에 나타나는지를 결정하는 보호 프로세스 리스트 검사단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 60.

제 58 항에 있어서, 상기 열기 포트요청의 상기 트랙킹은

프로세스 아이디와 상기 열기 포트요청에 대하여 리턴된 포트 핸들(handle)의 기록을 유지하는 단계로 이루어지고,

상기 닫기 포트의 트랙킹의 상기 닫기 포트 기록단계는 상기 기록으로부터 상기 프로세스의 상기 레코드와 상기 닫기 포트요청에 대하여 리턴된 포트 핸들을 제거하는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 61.

제 60 항에 있어서, 프로세스가 열기 포트를 가지는지를 검사하고, 열림 포트를 가진 프로세스에 대하여 보안해제를 거부하며, 열린 포트가 없는 프로세스에 대하여 보안해제를 허가하는 단계로 이루어진 보안검사단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 62.

제 61 항에 있어서, 상기 열기 포트 프로세스 검사단계는 상기 프로세스 동일성이 보호 프로세스 리스트상에 나타나는지를 결정하는 단계로 이루어지고, 열기 포트가 없는 프로세스에 대하여 보안해제를 허가하는 상기 단계는 상기 프로세스를 상기 보호 프로세스 리스트상에 위치시키는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 63.

제 5 항에 있어서, 내부보안을 제공하는 단계는

프로세스로 보내진 네트워크의 사용에 대하여 네트워크 요청을 감지하는 네트워크 요청 감지단계;

상기 요청 프로세스의 동일성을 결정하는 프로세스 확인단계;

상기 프로세스가 상기 네트워크를 엑세스하도록 허가하는지를 결정하는 프로세스 검사단계;

상기 프로세스가 상기 네트워크를 엑세스하도록 허가한다면 상기 네트워크 요청이 실행되도록 허가하고, 상기 프로세스가 상기 네트워크를 엑세스하도록 허가하지 않는다면 상기 네트워크 요청을 거부하는 허가/거부 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 64.

제 63 항에 있어서, 상기 프로세스 검사단계는 상기 프로세스가 보호 프로세스 리스트상에 나타나는지를 결정하는 보호 프로세스 리스트 검사단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 65.

제 64 항에 있어서, 상기 네트워크 요청 인터페이스가 전송 데이터 인터페이스인 것을 특징으로 하는 데이터보안 유지방법.

청구항 66.

제 1 항에 있어서, 패키지를 생성하는 단계는

패키징에 대하여 데이터 파일을 수신하는 단계;

상기 데이터 파일과 관련된 하나 이상의 승인을 가지는 승인 데이터베이스와, 상기 파일에 대하여 클라이언트의 사용을 결정하는 상기 하나 이상의 승인을 수신하는 단계;

패키지 전체의 유일한 식별자를 생성시키는 단계;

상기 파일과, 상기 하나이상의 승인, 상기 전체의 유일한 식별자로 이루어진 데이터의 패키지를 생성시키는 단계;

상기 패키지를 암호화하는 단계;

상기 암호화된 패키지로 이루어진 컴퓨터 실행파일을 생성시키는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 67.

제 66 항에 있어서, 상기 하나 이상의 승인은 엑세스 카운트 승인 , 엑세스 시간 승인, 만료 날짜 승인, 권한 날짜 승인, 클립보드 승인, 프린트 승인, 무제한 엑세스 승인, 애플리케이션 승인, 시스템-이벤트 승인으로 이루어진 그룹으로부터 선택되는 것을 특징으로 하는 데이터보안 유지방법.

청구항 68.

제 67 항에 있어서, 상기 컴퓨터 실행파일 에 대한 엑세스에 대해서 암호를 설정하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 69.

제 68 항에 있어서, 데이터의 상기 패키지는 수신자 전체의 유일한 식별자를 더 포함하고, 패키지 전체의 유일한 식별자를 생성시키는 단계 이후에 수신자 전체의 유일한 식별자를 수신하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 70.

제 69 항에 있어서, 데이터의 상기 패키지는 클라이언트 소프트웨어를 더 포함하여 구성된 것을 특징으로 하는 데이터 보안 유지방법.

청구항 71.

제 1 항에 있어서,

패키징에 대하여 데이터 파일을 수신하는 단계;

상기 데이터 파일과 관계되는 하나 이상의 승인을 가지는 패키지 승인 데이터베이스와 상기 파일의 클라이언트 사용자를 통제하는 하나 이상의 승인을 수신하는 단계;

패키지 전체의 유일한 식별자를 생성시키는 단계;

상기 데이터 파일, 상기 하나 이상의 승인, 상기 전체의 유일한 식별자, 클라이언트 소프트웨어로 이루어지는 데이터의 패키지를 생성시키는 단계;

상기 패키지를 암호화하는 단계;

상기 암호화된 패키지로 이루어지는 컴퓨터 실행파일을 생성시키는 단계;

운영체제를 가지는 클라이언트 컴퓨터에서 컴퓨터 실행파일을 수신하는 단계;

상기 운영체제가 호환성있는 운영체제인지를 결정하는 단계,

그렇다면 상기 클라이언트 컴퓨터 시스템에서 클라이언트 소프트웨어를 실행시키고 상기 클라이언트 소프트웨어의 실행이 클라이언트 승인 데이터베이스와 상기 클라이언트 컴퓨터 시스템상의 저장소를 생성시키는 단계,

상기 암호화된 패키지가 유효한지를 결정하는 단계,

유효하다면 상기 패키지 전체의 유일한 식별자를 상기 클라이언트 승인 데이터베이스에 기록하는 단계,

상기 데이터의 패키지로부터 상기 데이터 파일과 하나 이상의 승인을 추출하는 단계,

상기 데이터 파일을 상기 저장소에 저장하고 상기 하나 이상의 승인을 상기 클라이언트 승인 데이터베이스에 저장하는 단계,

유효하지 않다면 상기 패키지가 설치된 것을 나타내는 상기 컴퓨터 실행파일에서 상태를 설정하는 단계로 이루어진 상기 클라이언트 컴퓨터 시스템에서 상기 컴퓨터 실행파일을 실행시키는 단계; 를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 72.

제 71 항에 있어서, 두번째 패키지가 상기 컴퓨터 시스템에 로드되었는지를 결정하는 단계;

그렇다면, 상기 클라이언트 컴퓨터 시스템에서 클라이언트 소프트웨어를 실행시키는 단계 이전에 상기 두번째 패키지를 종결짓는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 73.

제 72 항에 있어서, 상기 패키지가 유효한지를 결정하는 단계는,

상기 패키지 전체의 유일한 식별자에 대하여 상기 클라이언트 승인 데이터베이스를 찾는단계;

상기 패키지 전체의 유일한 식별자가 상기 클라이언트 승인 데이터베이스상에 있지 않다면 상기 패키지가 유효하다고 결정하는 단계;

상기 패키지 전체의 유일한 식별자가 상기 클라이언트 승인 데이터베이스상에 없다면 상기 패키지가 유효하지 않다고 결정하는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 74.

제 73 항에 있어서, 상기 패키지는,

버전 지정을 가지는 상기 클라이언트 소프트웨어와;

상기 클라이언트 소프트웨어를 실행시키는 단계 이전에 상기 소프트웨어의 두번째 버전이 상기 클라이언트 시스템에 설치되었는지를 결정하는 단계;

그렇다면, 상기 패키지로부터 상기 클라이언트 소프트웨어를 추출하고 상기 클라이언트 컴퓨터 시스템에 상기 클라이언트 소프트웨어를 설치하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 75.

제 74 항에 있어서, 상기 클라이언트 소프트웨어의 두번째 버전이 상기 클라이언트 컴퓨터 시스템에 설치된다면,

상기 상기 클라이언트 컴퓨터 시스템에 설치된 상기 클라이언트 소프트웨어의 상기 버전 지정이 두번째 버전보다 빠른 것인지를 결정하는 단계;

그렇다면, 상기 패키지로부터 상기 클라이언트 소프트웨어를 추출하고 상기 클라이언트 컴퓨터 시스템에 설치하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 76.

제 74 항에 있어서, 상기 클라이언트 소프트웨어는 하나 이상의 디바이스 드라이버와 상기 클라이언트 승인 데이터베이스로 이루어지고, 상기 저장소는 하나 이상의 디바이스 드라이버 중 적어도 하나에 의해서 생성되는 것을 특징으로 하는 데이터보안 유지방법.

청구항 77.

제 71 항에 있어서, 상기 클라이언트 소프트웨어는 하나 이상의 디바이스 드라이버와 상기 클라이언트 승인 데이터베이스로 이루어지고, 상기 저장소는 하나 이상의 디바이스 드라이버 중 적어도 하나에 의해서 생성되는 것을 특징으로 하는 데이터보안 유지방법.

청구항 78.

제 71 항에 있어서, 상기 패키지는 수신자 전체의 유일한 식별자를 더 포함하고, 상기 암호화된 패키지가 유효한지를 결정하는 단계가 두번째 수신자 전체의 유일한 식별자에 대하여 상기 클라이언트 승인 데이터베이스를 찾는 단계를 포함하며, 찾지 못한다면 상기 패키지가 유효하지 않은 것으로 결정짓고, 찾는다면 상기 수신자 전체의 유일한 식별자를

상기 두번째 수신자 전체의 유일한 식별자와 비교하여 일치되는지를 결정하며, 일치한다면 상기 패키지가 유효한 것으로 결정짓고, 일치하지 않는다면 상기 패키지가 유효하지 않다고 결정짓는 것을 특징으로 하는 데이터보안 유지방법.

청구항 79.

제 71 항에 있어서, 상기 하나 이상의 승인은 엑세스 카운트 승인, 엑세스 시간 승인, 만료 날자 승인, 권한 날짜 승인, 클립보드 승인, 프린트 승인, 무제한 엑세스 승인, 애플리케이션 승인, 시스템-이벤트 승인으로 이루어진 그룹으로부터 선택되는 것을 특징으로 하는 데이터보안 유지방법.

청구항 80.

제 71 항에 있어서, 상기 컴퓨터 실행파일이 보호된 암호인 것을 특징으로 하는 데이터보안 유지방법.

청구항 81.

제 5 항에 있어서, 내부보안을 제공하는 단계는

파일시스템 요청을 감지하는 단계;

상기 파일시스템 요청을 완료하는 단계;

상기 파일시스템 요청으로부터 리턴 정보를 수신하는 단계;

상기 파일시스템 요청이 보호 파일과 관련된 태그파일에 관한 것인지를 결정하는 단계;

그렇다면, 상기 보호 파일의 파일 속성을 나타내는 상기 리턴 정보를 변경하는 단계로 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 82.

제 81 항에 있어서, 상기 파일 속성이 파일크기인 것을 특징으로 하는 데이터보안 유지방법.

청구항 83.

제 81 항에 있어서, 결정하는 단계가,

상기 리턴 정보가 다수의 보호 파일과 관련된 다수의 태그파일과 동일한지를 결정하는 단계;

그렇다면, 상기 다수의 보호 파일의 파일 속성을 나타내는 상기 리턴 정보를 변경하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지방법.

청구항 84.

제 81 항에 있어서, 상기 보호 파일이 암호화된 형태로 저장되는 것을 특징으로 하는 데이터보안 유지방법.

청구항 85.

제 81 항에 있어서, 상기 보호 파일이 보호 가상 파일시스템에 저장되는 것을 특징으로 하는 데이터보안 유지방법.

청구항 86.

제 81 항에 있어서, 상기 보호 파일이 먼 네트워크에 연결된 디바이스에 저장된 것을 특징으로 하는 데이터보안 유지방법.

청구항 87.

제 81 항에 있어서, 상기 파일시스템 요청은 파일 열기인 것을 특징으로 하는 데이터보안 유지방법.

청구항 88.

제 81 항에 있어서, 상기 파일시스템 요청은 파일 삭제인 것을 특징으로 하는 데이터보안 유지방법.

청구항 89.

제 81 항에 있어서, 상기 파일시스템 요청은 파일 이름바꾸기인 것을 특징으로 하는 데이터보안 유지방법.

청구항 90.

제 81 항에 있어서, 상기 파일시스템 요청은 파일 정보에 대한 질문인 것을 특징으로 하는 데이터보안 유지방법.

청구항 91.

제 83 항에 있어서, 상기 파일시스템 요청은 파일 정보 설정인 것을 특징으로 하는 데이터보안 유지방법.

청구항 92.

제 83 항에 있어서, 상기 파일시스템 요청은 첫번째 일치하는 파일 찾기인 것을 특징으로 하는 데이터보안 유지방법.

청구항 93.

제 83 항에 있어서, 상기 파일시스템 요청은 다음 일치하는 파일 찾기인 것을 특징으로 하는 데이터보안 유지방법.

청구항 94.

제 83 항에 있어서, 상기 파일시스템 요청은 디렉토리 제어인 것을 특징으로 하는 데이터보안 유지방법.

청구항 95.

데이터 사용을 통제하기 위하여 상기 데이터와 하나 이상의 승인으로 이루어진 패키지를 처리하기 위한 리시버;

상기 데이터를 저장하기 위한 저장소로 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 96.

제 95 항에 있어서, 상기 저장소에 저장되는 상기 데이터를 보호하기 위하여 내부보안을 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 97.

제 96 항에 있어서, 상기 내부보안이 상기 하나 이상의 승인의 침입을 감지하는 것을 특징으로 하는 데이터보안 유지시스템.

청구항 98.

제 97 항에 있어서, 상기 내부보안은 상기 데이터와 일치하는 태그파일과, 상기 데이터에 대한 실제 파일이름과 상기 태그파일에 대한 태그이름을 사용하므로서 상기 데이터에 대항하는 상기 태그파일을 맵핑하는 가상 테이블로 이루어지고, 상기 가상 테이블과 데이터는 상기 저장소에 저장되는 것을 특징으로 하는 데이터보안 유지시스템.

청구항 99.

제 97 항에 있어서, 상기 내부보안은 적어도 하나의 상기 저장소에 대하여 최초의 위치와 일치하는 고정 어드레스, 상기 승인을 저장하는 상기 패키지와 데이터베이스의 읽기위해 사용되는 드라이버로 이루어지고,

시스템 작동의 통제에 대하여 키를 제공하기 위하여 상기 어드레스를 결합시키고 상기 키가 작동하지 않을 때를 확인하는 것을 특징으로 하는 데이터보안 유지시스템.

청구항 100.

제 97 항에 있어서, 상기 내부보안은,

호출 프로세스에서 레지스트리 키에 대한 핸들;

상기 핸들에 대한 레지스트리 키값;

프로세스 아이디와 레지스트리 키;

상기 요청을 완료시키기 위한 보안해제;

보호 프로세스 리스트를 검사하므로서 보호되는 프로세스;

상기 프로세스가 보호된다면, 상기 레지스트리 키가 거절 리스트상에 있는지를 결정하는 단계;

상기 레지스트리 키가 상기 거절 리스트상에 있다면, 상기 레지스트리 키에 대한 상기 프로세스 엑세스를 거부하는 단계;

상기 프로세스가 상기 보호 피스트상에 없거나 상기 레지스트리 키이름이 상기 거절 리스트상에 없다면, 상기 요청을 완료하는 간계로 이루어진 시스템 감시 레지스트리를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 101.

제 97 항에 있어서, 상기 내부보안은

요청 프로세스에 대하여 메모리 페이지를 예약하기 위한 요출;

상기 페이지가 공유될 수 있는지에 따라 필터되는 예약호출;

요청 프로세스나 다음 프로세스에 대하여 상기 메모리 페이지를 위임하는 호출;

상기 페이지가 공유될 수 있는지 그리고 상기 프로세스가 보호될 수 있는지에 따라 필터되는 상기 위임 호출로 이루어진 공유메모리 시스템으로 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 102.

제 97 항에 있어서, 상기 내부보안은

다른 시스템 데이터로부터 저장 데이터를 분리시키기 위한 저장 시스템;

파일시스템 호출을 차단하는 파일시스템 보안드라이버;

상기 차단 파일시스템 호출 중 어떤 특정한 하나에 대하여,

상기 차단 파일시스템 호출 중 상기 특정한 하나가 상기 저장 데이터를 엑세스하는 프로세스로부터 인지를 결정하는 단계;

상기 차단 파일시스템 호출의 상기 특정한 하나가 상기 저장 데이터를 엑세스하는 프로세스로부터이라면, 상기 파일시스템이 상기 저장 시스템내에만 데이터를 생성하거나 변경하도록 허가하는 단계를 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 103.

제 97 항에 있어서, 상기 내부보안은,

시스템 시계로부터 첫번째 시간값을 읽는 단계;

하나 이상의 시계-관련의 승인 필드를 가지는 승인 데이터베이스가

컴퓨터 시스템상에서 초기화되는 저장된 시간값,

상기 승인 데이터베이스가 초기화되지 않는다면 상기 첫번째 시간값을 상기 저장된 시간값과 비교하며,

상기 처음 시간값이 상기 저장된 시간값보다 늦는다면 상기 첫번째 시간값을 상기 저장된 시간값 필드에 저장하고,

상기 처음 시간값이 상기 저장된 시간값보다 이르다면 상기 하나 이상의 시계-관련의 승인 기능을 억제하고,

그에 따라 상기 시계-관련의 승인이 상기 데이터에 대한 엑세스를 차단하는 기능을 억제하며,

상기 승인 데이터베이스가 초기화되지 않았다면, 상기 첫번째 시간값을 상기 저장된 시간값 필드에 저장하는 단계로 이루어지는 저장된 시간값 필드와, 하나 이상의 시계-관련의 승인으로 구성된 각각의 필드인지를 결정하는 단계;로 이루어진 데이터에 대한 권한없는 엑세스를 차단하기 위하여 컴퓨터의 시스템 시계를 감시하는 시스템을 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 104.

제 97 항에 있어서, 상기 내부보안은,

상기 첫번째 디바이스 드라이버에서 입출력 요청을 감지하는 단계;

상기 첫번째 디바이스 드라이버가 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위인지를 결정하는 단계;

상기 첫번째 디바이스 드라이버가 기능적으로 층을 이룬 다수의 디바이스 드라이버에서 최상위에 있다면, 상기 첫번째 디바이스 드라이버에서 상기 입출력 요청을 실행하는 단계;

상기 첫번째 디바이스 드라이버가 기능적으로 층을 이룬 다수의 디바이스 드라이버에서 최상위에 있지 않다면, 상기 첫번째 디바이스 드라이버에서 상기 입출력 요청을 거부하는 단계, 상기 층을 이룬 다수의 디바이스 드라이버에서 다음 하위-레벨 디바이스 드라이버에 의해 실행되도록 상기 입출력 요청을 허가하는 단계; 로 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 105.

제 97 항에 있어서, 상기 내부보안은,

프로세스에 의해 보내진 포트의 사용에 대하여 포트요청을 감지하는 포트요청 감지단계;

상기 요청 프로세스의 동일성을 결정하는 프로세스 확인단계;

상기 프로세스가 상기 포트에 대하여 엑세스하도록 허가되는지를 결정하는 프로세스 검사단계;

상기 프로세스가 상기 포트에 대하여 엑세스하도록 허가된다면 상기 포트요청이 실행되도록 허가되고 상기 프로세스가 상기 포트에 대하여 엑세스하도록 허가되지 않는다면 상기 포트요청을 거부하는 허가/거부 단계;로 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 106.

제 97 항에 있어서, 내부보안은

프로세스에 의해 보내진 포트의 사용에 대하여 포트요청을 검사하는 포트요청 검사단계;

상기 포트요청이 열기 포트요청이라면, 상기 요청 프로세스의 동일성을 결정하는 열기 포트 프로세스 확인단계;

상기 포트요청이 열기 포트요청이라면, 상기 프로세스가 상기 포트를 열도록 허가받는지를 결정하는 열기 포트 프로세스 검사단계;

상기 포트요청이 열기 포트요청이라면 상기 열기 포트요청이 실행되도록 허가하고, 상기 프로세스가 상기 포트를 열도록 허가되면 상기 열기 포트요청을 트랙킹하며, 상기 프로세스가 상기 포트를 열도록 허가되지 않는다면 상기 포트요청을 거부하는 열기 포트 허가/거부 단계;

상기 포트요청이 닫기 포트요청이라면 상기 포트요청을 완료하는 닫기 포트 프로세스 완료단계;

상기 포트의 닫기를 기록하는 닫기 포트 기록단계;로 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 107.

제 97 항에 있어서, 내부보안은

프로세스에 의해 보내진 네트워크의 사용에 대하여 네트워크 요청을 감지하는 네트워크 요청 감지단계;

상기 요청 프로세스의 동일성을 결정하는 프로세스 확인단계;

상기 프로세스가 상기 네트워크에 대하여 엑세스가 허가되는지를 결정하는 프로세스 검사단계;

상기 프로세스가 상기 네트워크를 엑세스하도록 허가된다면 상기 네트워크 요청이 실행되도록 허가하고 상기 프로세스가 상기 네트워크를 엑세스하도록 허가되지 않는다면 상기 네트워크 요청을 거부하는 허가/거부 단계;로 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 108.

제 96 항에 있어서,

데이터 파일;

상기 데이터 파일과 관련된 하나 이상의 승인을 가지는 승인 데이터베이스;

암호화 소프트웨어;

매체를 읽을 수 있는 기계와 통신상에 있는 네트워크;

상기 네트워크와 통신상에 있는 클라이언트 컴퓨터 시스템, 상기 정보의 패키지를 수신하고 컴퓨터 실행파일을 실행시키도록 적용되는 컴퓨터 시스템, 클라이언트 승인을 가지는 컴퓨터 시스템, 정보의 패키지를 수신하도록 적용되는 저장소;로 이루어지는 정보의 패키지로 이루어지는 컴퓨터 실행파일을 생성시키는 정보 패키징 소프트웨어를 가지는 매체를 읽을 수 있는 기계로 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 109.

제 108 항에 있어서, 상기 정보의 패키지는 패키지 전체의 유일한 식별자 더 포함하여 이루어지고, 상기 클라이언트 컴퓨터 시스템은 상기 패키지 전체의 유일한 식별자를 읽고 상기 패키지 전체의 유일한 식별자에 대하여 상기 클라이언트 승인 데이터베이스를 찾고 상기 패키지 전체의 유일한 식별자가 상기 클라이언트 승인 데이터베이스에서 찾아진다면 상기 패키지를 거절하도록 적용되는 컴퓨터 코드의 모듈을 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 110.

제 109 항에 있어서, 상기 정보의 패키지는 수신자 전체의 유일한 식별자를 도 포함하고, 상기 클라이언트 컴퓨터 시스템은 수신자 전체의 유일한 식별자를 읽고 상기 수신자 전체의 유일한 식별자에 대하여 상기 클라이언트 승인 데이터베이스를 찾고 상기 수신자 전체의 유일한 식별자가 상기 클라이언트 승인 데이터베이스에서 찾게되지 못한다면 상기 패키지를 거절하는 컴퓨터 코드의 모듈을 더 포함하여 구성된 것을 특징으로 하는 데이터보안 유지시스템.

청구항 111.

제 110 항에 있어서, 상기 하나 이상의 승인은 엑세스 카운트 승인, 엑세스 시간 승인, 만료 날짜 승인, 권한 날짜 승인, 클립보드 승인, 프린트 승인, 무제한 엑세스 승인, 애플리케이션 승인, 시스템-이벤트 승인으로 이루어진 그룹으로부터 선택되는 것을 특징으로 하는 데이터보안 유지시스템.

청구항 112.

제 97 항에 있어서, 상기 시스템은 데이터를 엑세스하는데 대하여 디바이스 드라이버를 포함하고, 상기 디바이스 드라이버가 전자 컴퓨터에 운영체제상에 설치되어 있으며,

상기 디바이스 드라이버는,

파일시스템 요청을 감지하고;

상기 파일시스템 요청을 완료하며;

상기 파일시스템 요청으로부터 리턴 정보를 수신하고;

상기 파일시스템 요청이 보호 파일과 관련된 태그파일에 대한 것인지를 결정하며;

그렇다면, 상기 보호 파일의 파일속성으로 나타내기위하여 상기 리턴 정보를 변경하는 것을 특징으로 하는 데이터보안 유지시스템.

청구항 113.

제 112 항에 있어서, 상기 파일 속성이 파일크기인 것을 특징으로 하는 데이터보안 유지시스템.

청구항 114.

제 113 항에 있어서, 상기 디바이스 드라이버는 상기 리턴 정보가 다수의 보호 파일과 관련된 다수의 태그파일을 확인할지를 결정하고, 그렇다면 상기 다수의 보호 파일의 파일 속성을 나타내기위한 상기 리턴 정보를 변경하는 것을 특징으로 하는 데이터보안 유지시스템.

청구항 115.

제 114 항에 있어서, 상기 첫번째 디바이스 드라이버는 파일시스템 감시자인 것을 특징으로 하는 데이터보안 유지시스템.

청구항 116.

제 114 항에 있어서, 상기 보호 파일이 암호화된 형태로 저장되는 것을 특징으로 하는 데이터보안 유지시스템.

청구항 117.

제 114 항에 있어서, 상기 보호 파일이 보호 가상 파일시스템에 저장되는 것을 특징으로 하는 데이터보안 유지시스템.

청구항 118.

제 114 항에 있어서, 상기 보호 파일은 이격되어 네트워크로 연결된 디바이스에 저장되는 것을 특징으로 하는 데이터보안 유지시스템.

청구항 119.

제 114 항에 있어서, 상기 파일시스템 요청은 파일 열기인 것을 특징으로 하는 데이터보안 유지시스템.

청구항 120.

제 114 항에 있어서, 상기 파일시스템 요청은 파일 삭제인 것을 특징으로 하는 데이터보안 유지시스템.

청구항 121.

제 114 항에 있어서, 상기 파일시스템 요청은 파일 이름바꾸기인 것을 특징으로 하는 데이터보안 유지시스템.

청구항 122.

제 114 항에 있어서, 상기 파일시스템 요청은 파일 정보 질문인 것을 특징으로 하는 데이터보안 유지시스템.

청구항 123.

제 114 항에 있어서, 상기 파일시스템 요청은 파일 정보 설정인 것을 특징으로 하는 데이터보안 유지시스템.

청구항 124.

제 123 항에 있어서, 상기 파일시스템 요청은 파일과 일치하는 파일 찾기인 것을 특징으로 하는 데이터보안 유지시스템.

청구항 125.

제 123 항에 있어서, 상기 파일시스템 요청은 다음 일치하는 파일 찾기인 것을 특징으로 하는 데이터보안 유지시스템.

청구항 126.

제 123 항에 있어서, 상기 파일시스템 요청은 디렉토리 제어인 것을 특징으로 하는 데이터보안 유지시스템.

청구항 127.

제 97 항에 있어서, 포트 블록킹 시스템을 더 포함하며,

상기 포트 블록킹 시스템은 프로세스에 의해 보내진 포트의 사용에 대하여 포트요청을 감지하도록 작동하고; 상기 요청 프로세스의 동일성을 결정하며; 상기 프로세스가 상기 포트에 대한 엑세스를 허가받는지를 결정하고;상기 프로세스가 상기 포트를 엑세스하도록 허가받는다면 상기 포트요청을 실행되도록 허가하며; 상기 프로세스가 상기 포트를 엑세스하도록 허가받지 못한다면 상기 포트요청을 거부하는 것을 특징으로 하는 데이터보안 유지시스템.

청구항 128.

제 97 항에 있어서, 네트워크 블록킹 시스템을 더 포함하며,

상기 네트워크 블록킹 시스템은 상기 요청 프로세스의 동일성을 결정하도록 작동하고; 상기 프로세스가 상기 네트워크에 대한 엑세스를 허가받을지를 결정하며; 상기 프로세스가 상기 네트워크에 대한 엑세스를 허가받는다면 상기 네트워크 요청을 실행시키도록 허가하고; 상기 프로세스가 상기 네트워크를 엑세스하도록 허가받지 못한다면 상기 네트워크 요청을 거부하는 것을 특징으로 하는 데이터보안 유지시스템.

청구항 129.

데이터로 이루어지는 패키지를 생성시키기 위한 적어도 하나 이상의 모듈과,

상기 패키지를 열고 상기 데이터의 제한된 엑세스에 대하여 저장소의 데이터를 저장하기 위한 상기 데이터와 모듈의 사용을 통제하는 하나 이상의 승인으로 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 130.

제 129 항에 있어서, 상기 하나 이상의 승인의 위반을 감지하는 모듈을 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 131.

제 130 항에 있어서, 상기 저장소에 저장되고 상기 데이터의 실제 파일이름과 상기 태그파일에 대하여 일치하는 태크이름을 포함하는 가상 테이블과 함께,

상기 저장소의 상기 데이터와 일치하는 태그파일을 생성시키고 가상 테이블의 상기 데이터에 대항하는 상기 태그파일을 맵핑하는 모듈을 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 132.

제 130 항에 있어서, 상기 승인을 저장하는 상기 패키지와 데이터베이스의 읽기, 운영체제를 통제하기 위한 키를 제공하기 위하여 어드레스와 결합, 상기 키가 작동되지 않을 때 확인하기 위하여 사용되는 드라이버와, 적어도 하나의 상기 저장소에 대하여 최초의 위치와 일치하는 고정 어드레스에 대한 확인을 포함하는 내부보안을 제공하기 위한 모듈을 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 133.

제 130 항에 있어서, 요청 프로세스에서 레지스트리 키에 대한 핸들을 요청하는 모듈;

상기 핸들에 대한 레지스트리 키값을 요청하는 모듈;

상기 요청을 완료하도록 보안해제를 획득하는 모듈;로 이루어진 레지스트리를 감시하는 모듈을 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 134.

제 130 항에 있어서, 요청 프로세스에 대하여 메모리 페이지를 예약하기 위하여 호출을 제공하는 모듈;

상기 페이지가 공유될 수 있는지에 따라 예약 호출을 필터링하는 모듈;

상기 요청 프로세스나 다음의 요청 프로세스에 대하여 상기 메모리 페이지를 위임하기 위한 호출을 제공하는 모듈;

상기 페이지가 공유될 수 있거나 상기 프로세스가 보호될 수 있는지에 따라 상기 위임 호출을 필터링하는 모듈;로 이루어지는 공유메모리를 감시하는 모듈을 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 135.

제 130 항에 있어서,

다른 시스템 데이터로부터 저장 데이터를 분리시키기 위한 저장 시스템을 제공하는 모듈;

파일시스템 호출을 차단하고, 차단 파일시스템 호출의 어떤 특정한 하나에 대하여 상기 차단 파일시스템 호출의 상기 특정한 하나의 것이 상기 저장 데이터를 엑세스하는 프로세스로부터인지를 결정하며, 상기 차단 파일시스템 호출의 상기 특정한 하나의 것이 상기 저장 데이터를 엑세스하는 프로세스로부터이라면 상기 파일시스템 호출이 상기 저장 시스템 내에서만 데이터를 생성 또는 변경하도록 허가하는 파일시스템 보호 드라이버를 제공하는 모듈;을 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 136.

제 130 항에 있어서,

상기 시스템 시계로부터 첫번째 시간값을 읽는 단계;

하나 이상의 시계-관련의 승인과 저장된 시간값으로 이루어지는 저장된 시간값 필드로 구성된 하나 이상의 시계-관련의 승인 필드를 가지는 승인 데이터베이스가 상기 컴퓨터 시스템에서 초기화되는지를 결정하는 단계;

상기 승인 데이터베이스가 초기화된다면, 상기 첫번째 시간값을 상기 저장된 시간값과 비교하고, 상기 첫번째 시간값이 상기 저장된 시간값보다 늦다면 상기 첫번째 시간값을 상기 저장된 시간값 필드에 저장하며, 상기 첫번째 시간값이 상기 저장된 시간값보다 이르다면 상기 하나 이상의 시계-관련의 승인 기능을 억제하고 그에 따라 상기 시계-관련의 스인이 상기 데이터에 대한 엑세스를 차단하는 기능을 억제하는 단계;

상기 승인 데이터베이스가 초기화되지 않는다면, 상기 첫번째 시간값이 상기 저장된 시간값 필드에 저장하는 단계;로 구성되는 시계 감시자를 초기화시키는 모듈:로 이루어지는 데이터에 대한 권한없는 엑세스를 차단하기 위하여 컴퓨터의 시스템 시계를 감시하는 모듈을 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 137.

제 130 항에 있어서,

상기 첫번째 디바이스 드라이버에서 입출력 요청을 감지하는 모듈;

상기 첫번째 디바이스 드라이버가 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위인지를 결정하는 모듈;

상기 첫번째 디바이스 드라이버가 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위에 있다면, 상기 첫번째 디바이스 드라이버에서 상기 입출력 요청을 실행시키는 모듈;

상기 첫번째 디바이스 드라이버가 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위에 있지 않다면, 상기 첫번째 디바이스 드라이버에서 상기 입출력 요청을 거부하고 상기 입출력 요청이 상기 층을 이룬 다수의 디바이스 드라이버에서 다음 하위-레벨 디바이스 드라이버에 의해서 실행되도록 허가하는 모듈;을 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 138.

제 129 항에 있어서, 프로세스에 의해 보내진 포트의 사용에 대하여 포트요청을 감지하는 포트요청 감지단계;

상기 요청 프로세스의 동일성을 결정하는 프로세스 확인단계;

상기 프로세스가 상기 포트를 엑세스하도록 허가받을지를 결정하는 프로세스 검사단계;

상기 프로세스가 상기 포트를 엑세스하도록 허가받는다면 상기 포트요청이 실행되도록 허가하고, 상기 프로세스가 상기 포트를 엑세스하도록 허가받지 못한다면 상기 포트요청을 거부하는 허가/거부 단계;를 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 139.

제 130 항에 있어서, 프로세스로부터 보내진 포트의 사용에 대하여 포트요청을 감지하는 포트요청 감지단계;

상기 포트요청이 열기 포트요청이라면 상기 요청 프로세스의 동일성을 결정하는 열기 포트 프로세스 확인단계;

상기 포트요청이 열기 포트요청이라면, 상기 프로세스가 상기 포트를 열도록 허가받는지를 결정하는 열기 포트 프로세스 검사단계;

상기 포트요청이 열기 포트요청이라면 상기 열기 포트요청이 실행되도록 허가하고, 상기 프로세스가 열기 포트를 열도록 허가받는다면 상기 열기 포트요청은 트랙킹하며, 상기 프로세스가 상기 포트를 열도록 허가받지 못한다면 상기 포트요청을 거부하는 열기 포트 허가/거부 단계;

상기 포트요청이 닫기 포트요청이라면, 상기 포트요청을 완료시키는 닫기 포트 프로세스 완료단계;

상기 포트의 닫기를 기록하는 닫기 포트 기록단계;를 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 140.

제 130 항에 있어서,

프로세스로부터 보내진 네트워크의 사용에 대하여 네트워크 요청을 감지하는 네트워크 요청 감지단계;

상기 요청 프로세스의 동일성을 결정하는 프로세스 확인단계;

상기 프로세스가 상기 네트워크에 대한 엑세스를 허가받는지를 결정하는 프로세스 검사단계;

상기 프로세스가 상기 네트워크에 대한 엑세스를 허가받는다면 상기 네트워크 요청이 실행되도록 허가하고 상기 프로세스가 상기 네트워크에 대한 엑세스를 허가받지 못한다면 상기 네트워크 요청을 거부하는 허가/거부 단계;를 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 141.

제 130 항에 있어서, 정보의 패키지는,

데이터 파일;

상기 데이터 파일과 관련된 하나 이상의 승인과 상기 파일의 클라이언트 사용을 통제하는 하나 이상의 승인을 가지는 승인 데이터베이스;

패키지 전체의 유일한 식별자;

수신자 전체의 유일한 식별자;로 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 142.

제 141 항에 있어서, 상기 하나 이상의 승인은 엑세스 카운트 승인, 엑세스 시간 승인, 만료 날짜 승인, 권한 날짜 승인, 클립보드 승인, 프린트 승인, 무제한 엑세스 승인, 애플리케이션 승인, 시스템-이벤트 승인으로 이루어지는 그룹으로부터 선택되는 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 143.

제 142 항에 있어서, 클라이언트 소프트웨어를 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 144.

제 130 항에 있어서, 데이터 엑세스를 위한 디바이스 드라이버 프로그램을 더 포함하며,

상기 디바이스 드라이버 프로그램은,

파일시스템 요청을 감지하는 컴퓨터-기반의 명령;

상기 파일시스템 요청을 완료하는 컴퓨터-기반의 명령;

상기 파일시스템 요청으로부터 리턴 정보를 수신하는 컴퓨터-기반의 명령;

상기 파일시스템 요청이 보호 파일과 관련된 태그파일에 대한 것인지를 결정하는 컴퓨터-기반의 명령;

상기 파일시스템 요청이 보호 파일과 관련된 태그파일에 대한 것이라면, 상기 보호 파일의 파일 속성을 나타내기 위하여 상기 리턴 정보를 변경하는 컴퓨터-기반의 명령;으로 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 145.

제 144 항에 있어서, 상기 디바이스 드라이버 프로그램은

상기 리턴 정보가 다수의 보호 파일과 관련된 다수의 태그파일과 동일한지를 결정하는 컴퓨터-기반의 명령;

상기 리턴 정보가 다수의 보호 파일과 관련된 다수의 태그파일과 일치한다면, 상기 다수의 보호 파일의 파일 속성을 나타내도록 상기 리턴 정보를 변경하는 컴퓨터-기반의 명령;을 더 포함하여 구성된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 146.

제 130 항에 있어서, 프로세스에 보내진 포트의 사용에 대하여 포트요청을 감지하도록 작동하고;

상기 요청 프로세스의 동일성을 결정하며;

상기 프로세스가 상기 포트를 엑세스하도록 허가받는지를 결정하고;

상기 프로세스가 상기 포트를 엑세스하도록 허가받는다면 상기 포트요청이 실행되도록 허가하고, 상기 프로세스가 상기 포트를 엑세스하도록 허가받지 못한다면 상기 포트요청을 거부하는; 포트 블록킹 시스템을 기반으로 하는 보호 데이터를 보호하도록 프로그램된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 147.

제 130 항에 있어서,

상기 요청 프로세스의 동일성을 결정하도록 작동되고;

상기 프로세스가 상기 네트워크를 엑세스하도록 허가받는지를 결정하며;

상기 프로세스가 상기 네트워크를 엑세스하도록 허가받는다면 상기 네트워크 요청이 실행되도록 허락하고, 상기 프로세스가 상기 네트워크를 엑세스하도록 허가받지않는다면 상기 네트워크 요청을 거부하는; 네트워크 블록킹 시스템을 기반으로 하는 보호 데이터를 보호하도록 프로그램된 것을 특징으로 하는 컴퓨터-읽기 매체.

청구항 148.

적어도 두 대의 컴퓨터 사이에서 데이터를 전송하는 동안,

상기 데이터의 사용 통제에 대하여 실행가능한 승인의 리스트로부터 선택된 하나 이상의 승인과 데이터로 이루어진 패키지를 생성시키는 시스템을 가진 첫번째 컴퓨터;

상기 첫번째 컴퓨터로부터 상기 패키지를 수신하고, 확인 후 상기 패키지를 열고, 상기 데이터를 저장소에 저장하는 시스템을 가진 두번째 컴퓨터;로 구성된 것을 특징으로 하는 보안 유지시스템.

청구항 149.

제 148 항에 있어서, 내부보안을 더 포함하며,

상기 내부보안은,

상기 하나 이상의 승인의 위반을 감지하는 단계;

상기 가상 테이블과 상기 데이터가 상기 저장소에 저장된다는 점에서, 상기 데이터와 일치하는 태그파일을 생성하고 가상 테이블에서 상기 데이터에 대항하는 상기 태그파일을 맵핑하며, 상기 가상 테이블은 상기 데이터의 실제 파일이름과 상기 태그파일에 대하여 일치하는 태그이름을 포함하고, 그 점에서 상기 가상 테이블과 상기 데이터가 상기 저장소에 저장되는 단계;

상기 패키지와 상기 승인을 저장하는 데이터베이스를 읽고 시스템 작동을 통제하는데 대한 키를 제공하기 위하여 어드레스를 함께 결합시키며 키가 작동하지 않을 때 확인하하는데 사용되는 드라이버와, 적어도 하나 의 상기 저장소에 대한 최초의 위치와 일치하는 고정 어드레스를 확인하는 단계;

호출 프로세스에서 레지스트리 키에 대하여 핸들을 요청하는 단계,

상기 핸들에 대하여 레지스트리 키값을 요청하는 단계,

상기 요청을 완료하도록 보안해제를 획득하는 단계로 이루어지는 레지스트를 감시하는 단계;

요청 프로세스에 대하여 메모리 페이지를 예약하기 위하여 호출을 제공하는 단계,

상기 페이지가 공유될 수 있는지에 따라 상기 예약 호출을 필터링하는 단계,

상기 요청 프로세스나 다음 요청 프로세스에 대하여 상기 메모리 페이지를 위임하기 위하여 호출을 제공하는 단계,

상기 페이지게 공유될 수 있는지 그리고 상기 프로세스가 보호될 수 있는지에 따라 상기 위임 호출을 필터링하는 단계로 이루어진 공유메모리 감시단계;

다른 시스템 데이터로부터 저장 데이터를 분리시키기 위하여 저장 시스템을 제공하는 단계;

파일시스템 호출을 차단하고, 상기 차단 파일시스템 호출의 어떤 특정한 하나에 대해서 상기 차단 파일시스템 호출의 상기 특정한 하나가 상기 저장 데이터를 엑세스하는 프로세스로부터인지를 결정하는 파일시스템 보안드라이버를 제공하고, 그리고 상기 차단 파일시스템 호출의 상기 특정한 하나가 상기 저장 데이터를 엑세스하는 프로세스로부터이라면 상기 파일시스템호출이 상기 저장 시스템내에서만 데이터를 생성 또는 삭제할 수 있도록 허가하는 단계;

상기 시스템 시계로부터 첫번째 시간값을 읽는 단계,

하나 이상의 시계-관련의 승인과 저장된 시간값을 포함하는 저장된 시간값 필드로 이루어지는 하나 이상의 시계-고나련의 승인 필드를 가지는 승인 데이터베이스가 상기 컴퓨터 시스템상에서 초기화되는지를 결정하는 단계,

상기 승인 데이터베이스가 초기화된다면 상기 첫번째 시간값을 상기 저장된 시간값과 비교하는 단계,

상기 첫번째 시간값이 상기 저장된 시간값보다 늦다면, 상기 첫번째 시간값을 상기 저장된 시간값 필드에 저장하는 단계,

상기 첫번째 시간값이 상기 저장된 시간값보다 이르다면 상기 정어도 하나의 실계-관련의 승인 기능을 억제하고 그 점에서 상기 시계-관련의 승인이 상기 데이터를 엑세스하는 기능을 억제하는 단계,

상기 승인 데이터베이스가 초기화되지 않는다면, 상기 첫번째 시간값을 상기 저장된 시간값 필드에 저장하는 단계로 이루어지는 시계 감시자를 초기화하는 단계로 구성되는 데이터에 대하여 권한없는 엑세스를 차단하기 위하여 컴퓨터 시스템 시계를 감시하는 단계;

상기 첫번째 디바이스 드라이버에서 입출력 요청을 검사하는 단계;

상기 첫번째 디바이스 드라이버가 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위인지를 결정하는 단계;

상기 첫번째 디바이스 드라이버가 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위에 있다면, 상기 첫번째 디바이스 드라이버에서 상기 입출력 요청을 실행하는 단계;

상기 첫번째 디바이스 드라이버가 기능적으로 상기 층을 이룬 다수의 디바이스 드라이버에서 최상위에 있지 않다면 상기 첫번째 디바이스 드라이버에서 상기 입출력 요청을 거부하고, 상기 입출력 요청이 상기 층을 이룬 다수의 디바이스 드라이버에서 다음 하위-레벨 디바이스 드라이버에 의해서 실행되도록 허가하는 단계;

프로세스에 의해 보내진 포트의 사용에 대하여 포트요청을 감지하는 단계;

상기 요청 프로세스의 동일성을 결정하는 단계;

상기 프로세스가 상기 포트를 엑세스하도록 허가받는지를 결정하는 단계;

상기 프로세스가 상기 포트를 엑세스하도록 허가받는다면 상기 포트요청이 실행되도록 허가하고, 상기 프로세스가 상기 포트를 엑세스하도록 허가받지 않는다면 상기 포트요청을 거부하는 단계;

프로세스에 의해서 보내진 포트의 사용에 대하여 포트요청을 감지하는 단계;

상기 포트요청이 열기 포트요청이라면, 상기 요청 프로세스의 동일성을 결정하는 단계;

상기 포트요청이 열기 포트요청이라면, 상기 프로세스가 상기 포트를 열도록 허가받는지를 결정하는 단계;

상기 포트요청이 열기 포트요청이라면 상기 열기 포트요청이 실행되도록 허가하고, 상기 프로세스가 상기 포트를 열도록 허가받는다면 상기 열기 포트요청을 트랙킹하며, 상기 프로세스가 상기 포트를 열도록 허가받지 않는다면 상기 포트요청을 거부하는 단계;

상기 포트요청이 닫기 포트요청이라면, 상기 포트요청을 완료하는 단계;

상기 포트의 닫기를 기록하는 단계;

프로세스로부터 보내진 네트워크의 사용에 대하여 네트워크 요청을 감지하는 단계;

상기 요청 프로세스의 동일성을 결정하는 단계;

상기 프로세스가 상기 네트워크를 엑세스하도록 허가받는지를 결정하는 단계;

상기 프로세스가 상기 네트워크를 엑세스하도록 허가받는다면 상기 네트워크 요청을 실행되도록 허가되고, 상기 프로세스가 상기 네트워크를 엑세스하도록 허가받지 않는다면 상기 네트워크 요청을 거부하는 단계;

파일시스템 요청을 감지하는 단계; 상기 파일시스템 요청을 완료하는 단계;

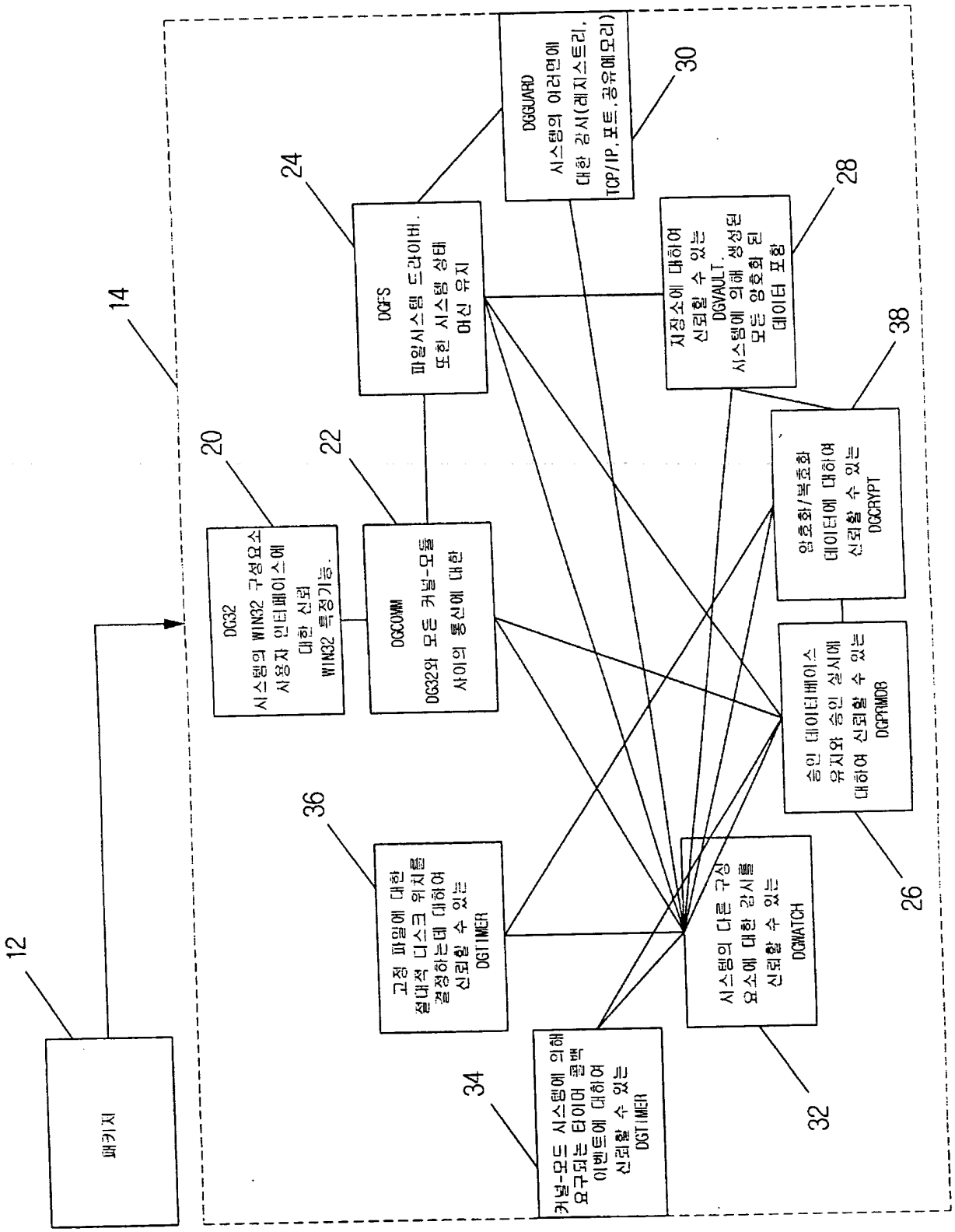
상기 파일시스템 요청으로부터 리턴 정보를 수신하는 단계;

상기 파일시스템 요청이 보호 파일과 관련된 태그파일에 관한 것인지를 결정하는 단계;

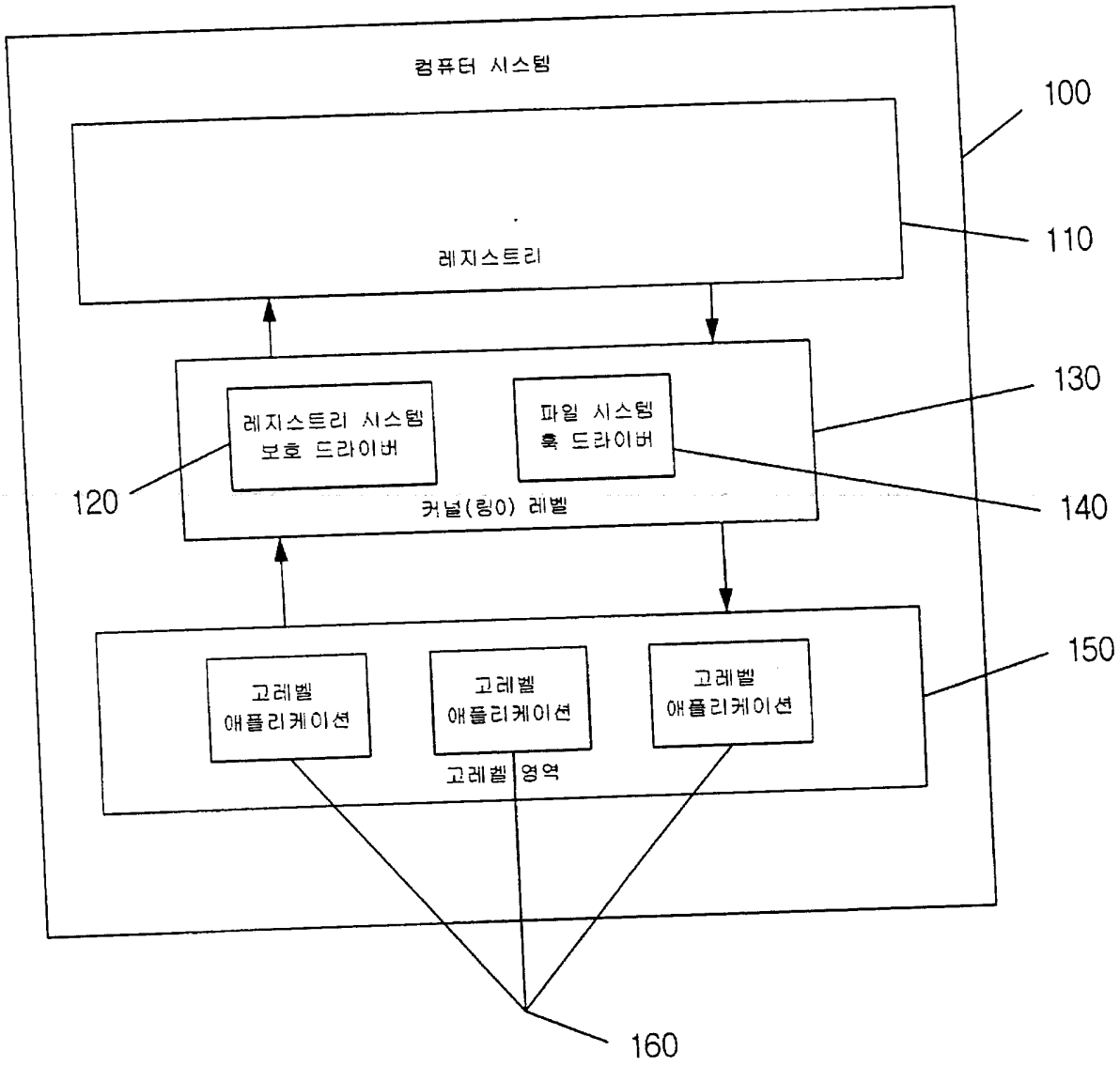
그렇다면, 상기 보호 파일의 파일 속성을 나타내기 위하여 상기 리턴 정보를 변경하는 단계;로 구성된 것을 특징으로 하는 보안 유지시스템.

도면

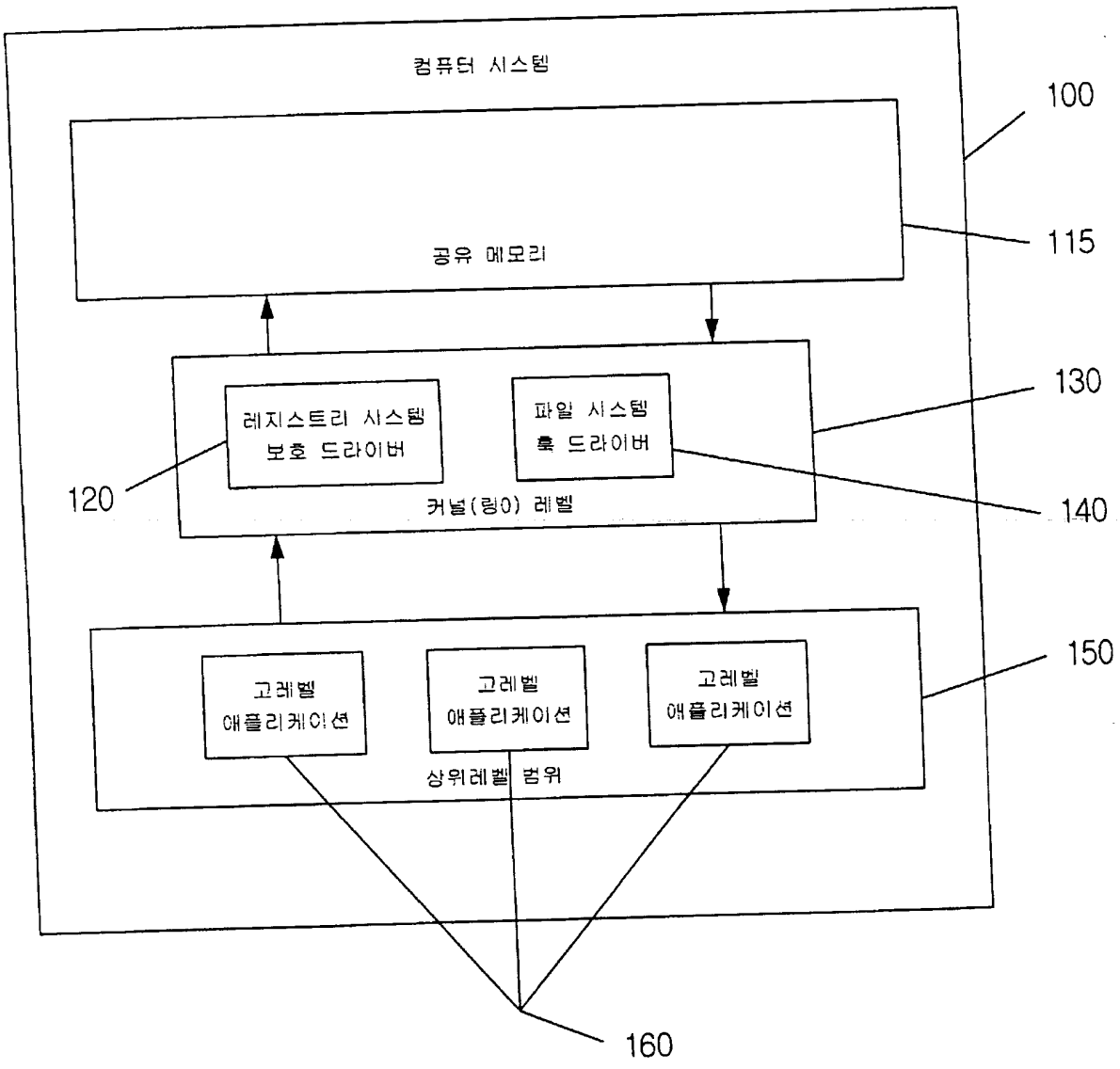
도면 1


12
패키지
14
20
DG32
시스템의 WIN32 구성요소
사용자 인터페이스에
대한 신뢰
WIN32 특정기능.
22
DGCOMM
DG32와 모든 커널-모듈
사이의 통신에 대한
24
DGFS
파일시스템 드라이버.
또한 시스템 상태
여신 유지
30
DGGUARD
시스템의 여러면에
대한 감시(레지스트리,
TCP/IP,포트,공유메모리)
28
저장소에 대하여
신뢰할 수 있는
DGVAULT.
시스템에 의해 생성된
모든 암호화 된
데이터 포함
38
암호화/복호화
데이터에 대하여
신뢰할 수 있는
DGCRYPT
26
승인 데이터베이스
유지와 승인 실시에
대하여 신뢰할 수 있는
DGPRMDB
32
시스템의 다른 구성
요소에 대한 감시를
신뢰할 수 있는
DGWATCH
34
커널-모드 시스템에 의해
요구되는 타이머 콜백
이벤트에 대하여
신뢰할 수 있는
DGTIMER
36
고정 파일에 대한
섹대적 디스크 위치를
결정하는데 대하여
신뢰할 수 있는
DGTIMER

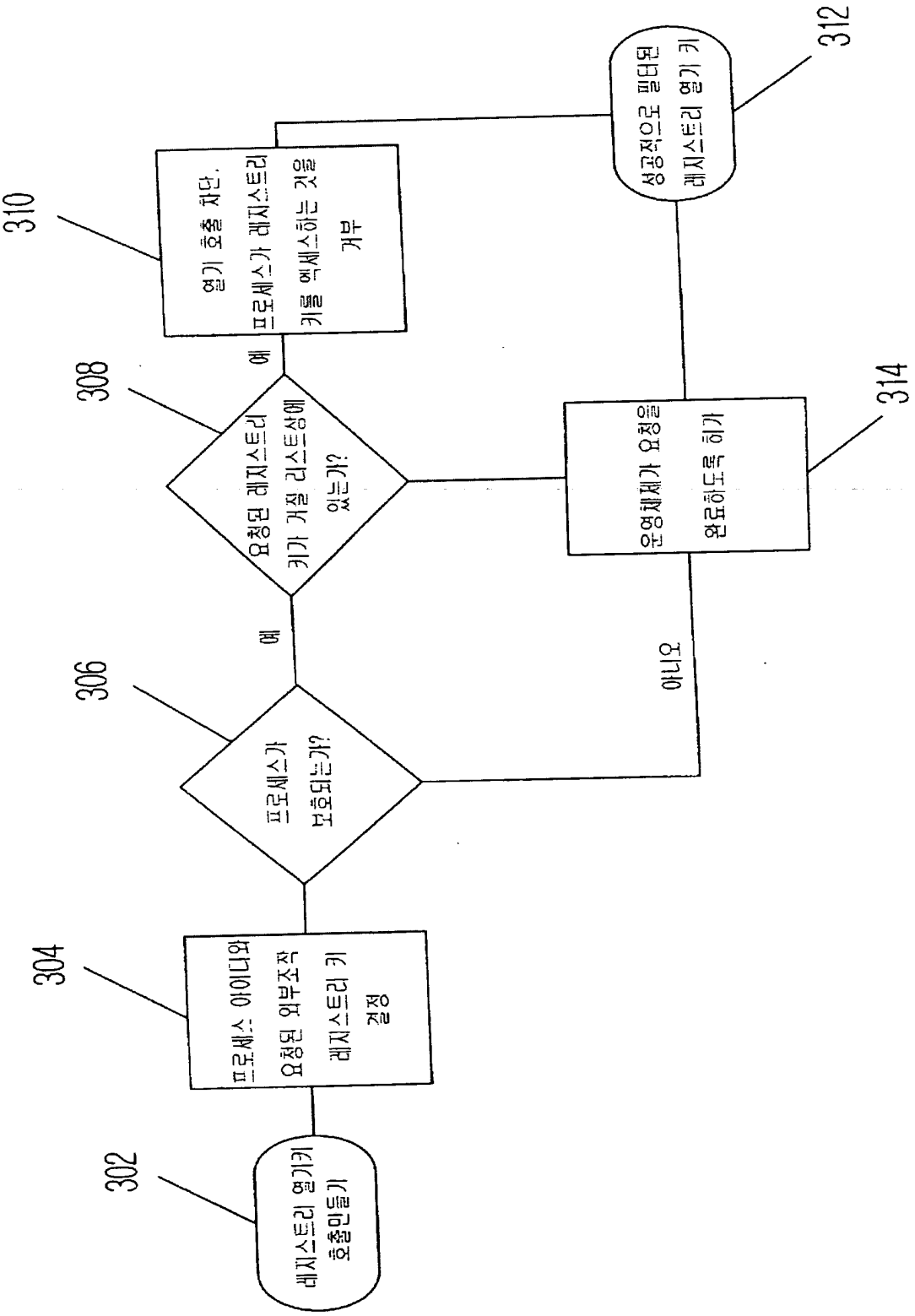
도면 2


컴퓨터 시스템
100
레지스트리
110
130
레지스트리 시스템
보호 드라이버
파일 시스템
훅 드라이버
120
140
커널(링0) 레벨
150
고레벨
애플리케이션
고레벨
애플리케이션
고레벨
애플리케이션
고레벨 영역
160

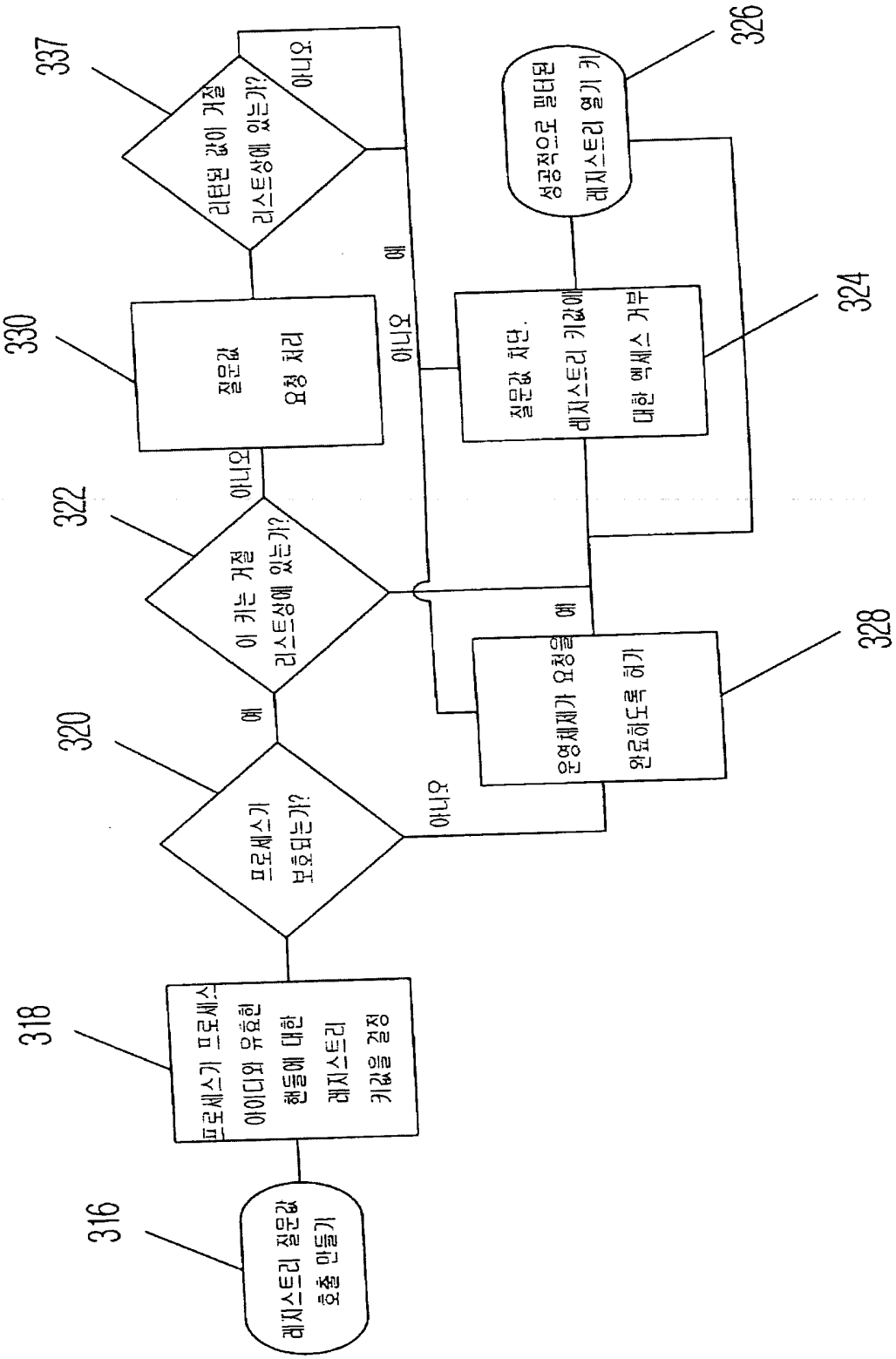
도면 3


컴퓨터 시스템
100
공유 메모리
115
레지스트리 시스템
보호 드라이버
파일 시스템
훅 드라이버
130
120
140
커널(링0) 레벨
고레벨
애플리케이션
고레벨
애플리케이션
고레벨
애플리케이션
150
상위레벨 컴위
160

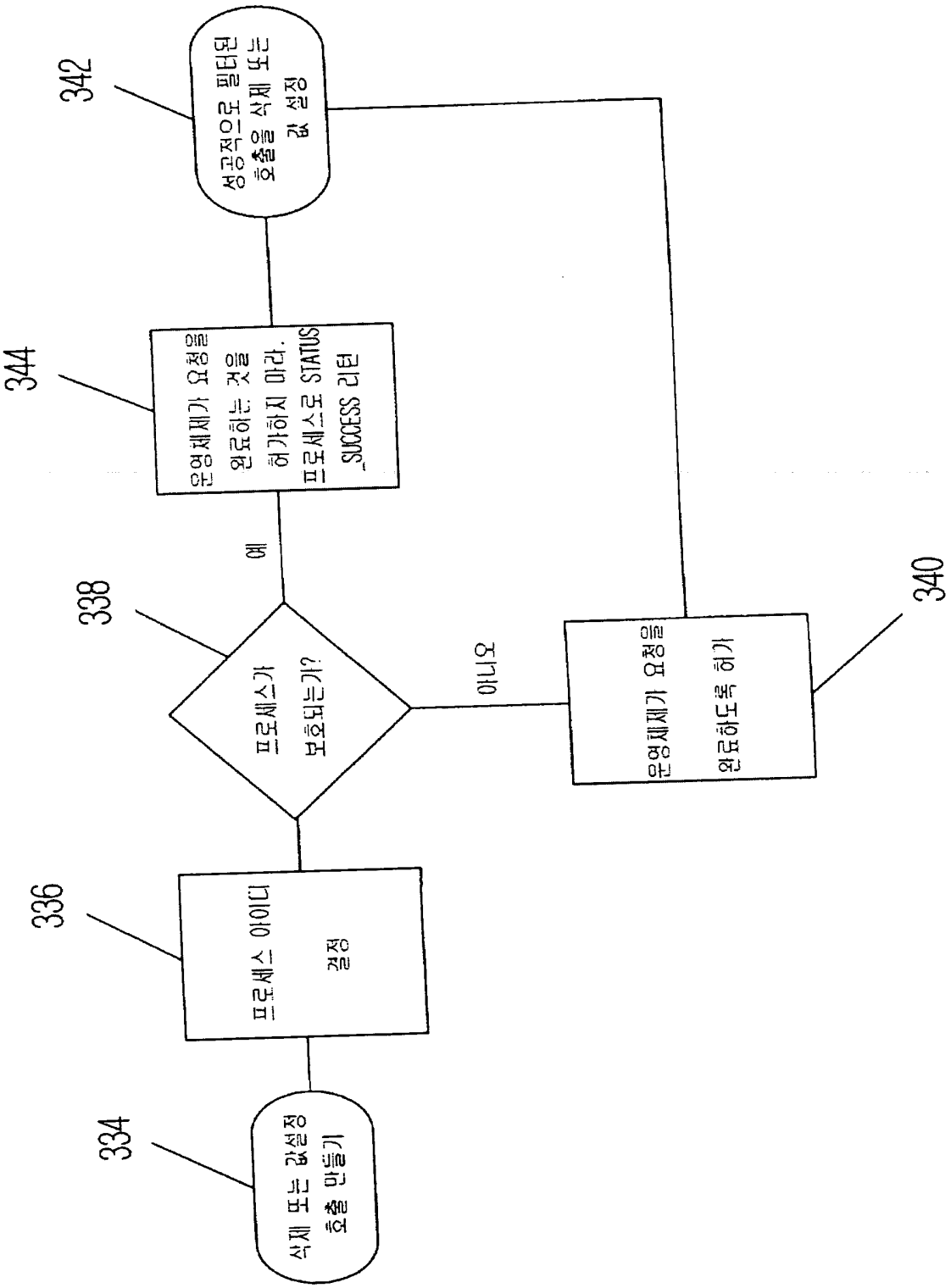
도면 4a


302
레지스트리 열기키 호출만들기
304
프로세스 아이디와 요청된 외부조작 레지스트리 키 결정
306
프로세스가 보호되는가?
예
아니오
308
요청된 레지스트리 키가 거절 리스트상에 있는가?
예
310
열기 호출 차단, 프로세스가 레지스트리 키를 엑세스하는 것을 거부
312
성공적으로 필터된 레지스트리 열기 키
314
운영체제가 요청을 완료하도록 허가

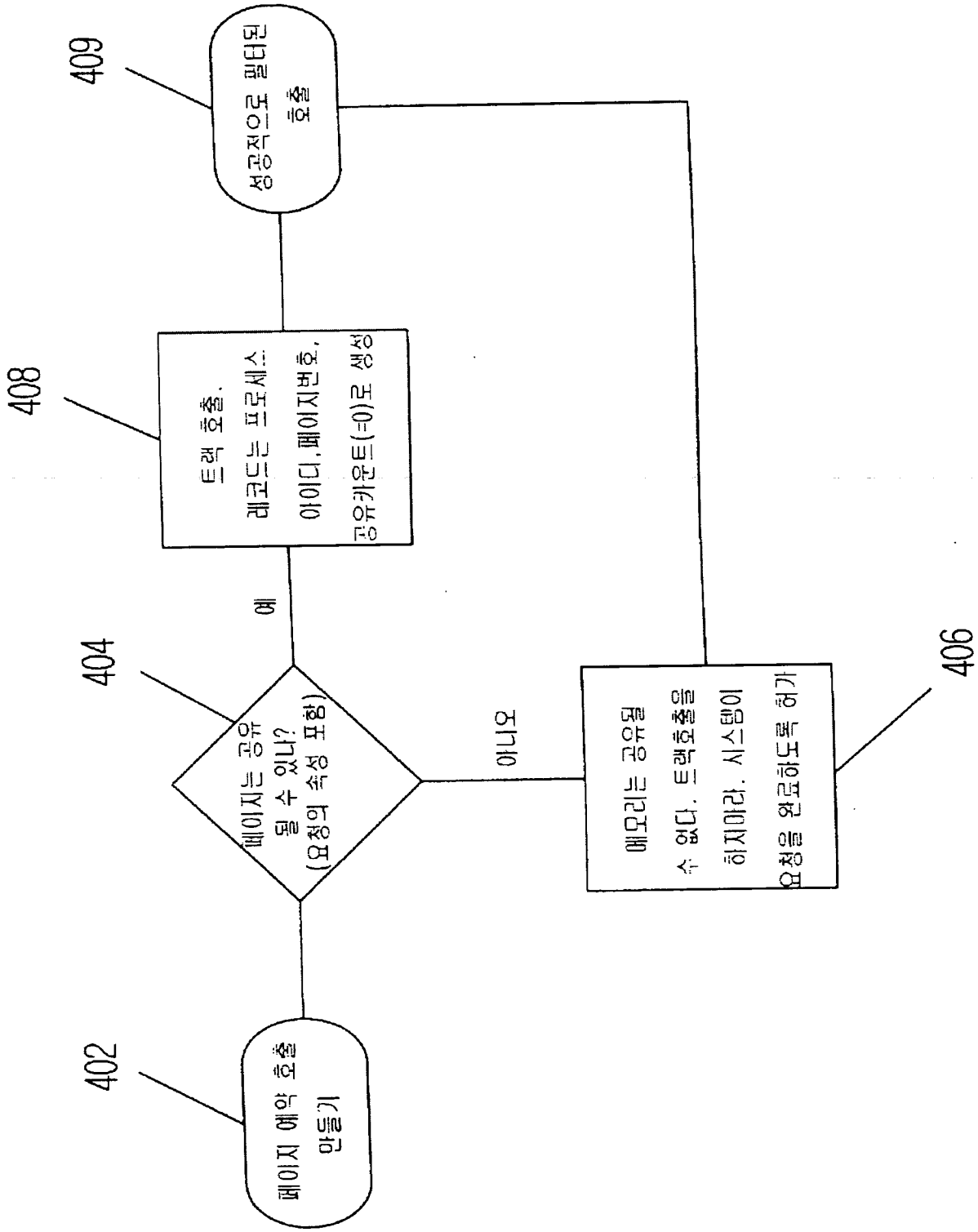
도면 4b


316
레지스트리 질문값 호출 만들기
318
프로세스가 프로세스 아이디와 유효한 핸들에 대한 레지스트리 키값을 결정
320
프로세스가 보호되는가?
예
아니오
322
이 키는 거절 리스트상에 있는가?
아니오
330
질문값 요청 처리
337
리턴된 값이 거절 리스트상에 있는가?
아니오
예
324
질문값 차단. 레지스트리 키값에 대한 엑세스 거부
326
성공적으로 필터된 레지스트리 열기 키
328
운영체제가 요청을 완료하도록 허가
예

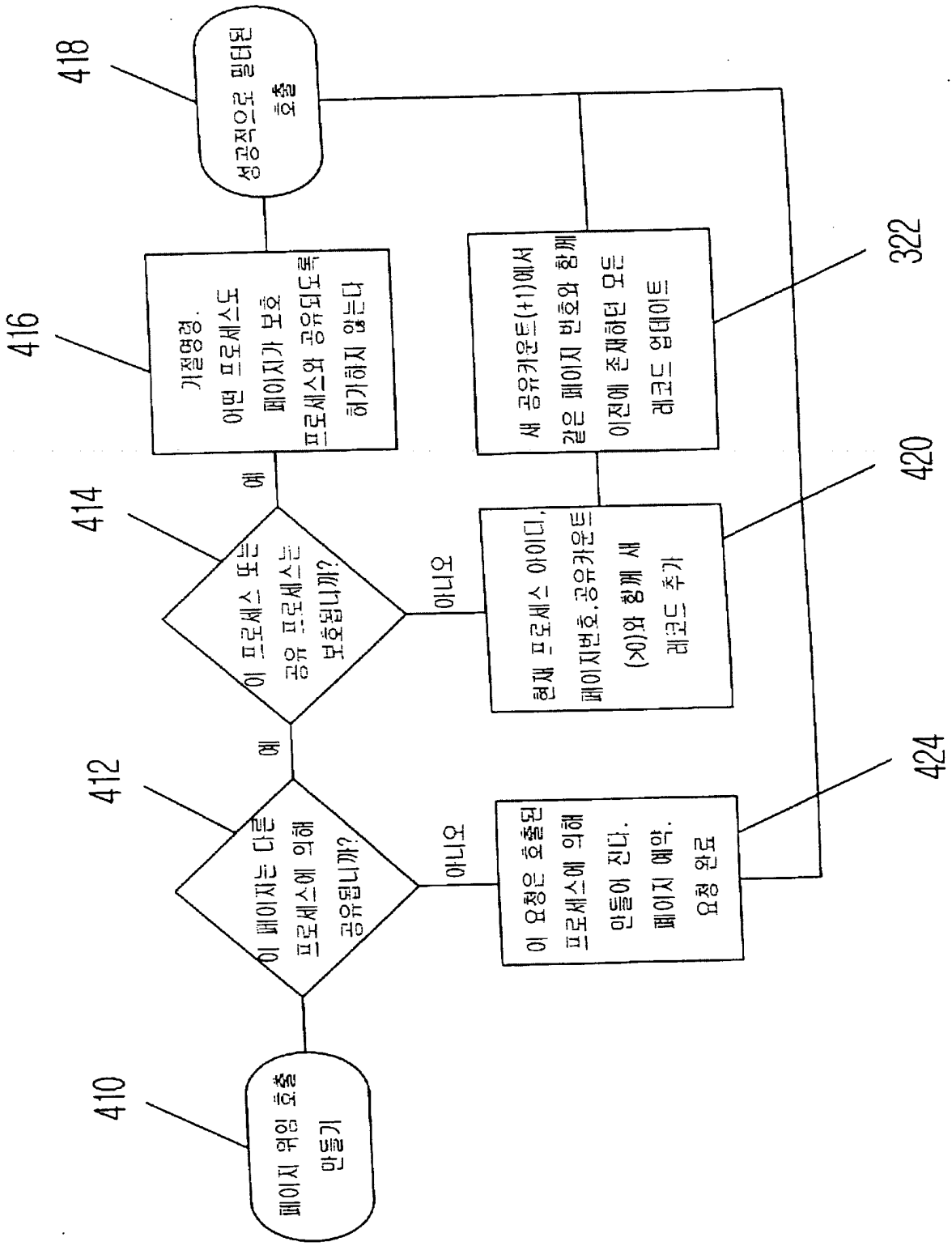
도면 4c


342
성공적으로 필터된 호출을 삭제 또는 값 설정
344
운영체제가 요청을 완료하는 것을 허가하지 마라. 프로세스로 STATUS_SUCCESS 리턴
예
338
프로세스가 보호되는가?
아니오
340
운영체제가 요청을 완료하도록 허가
336
프로세스 아이디 결정
334
삭제 또는 값설정 호출 만들기

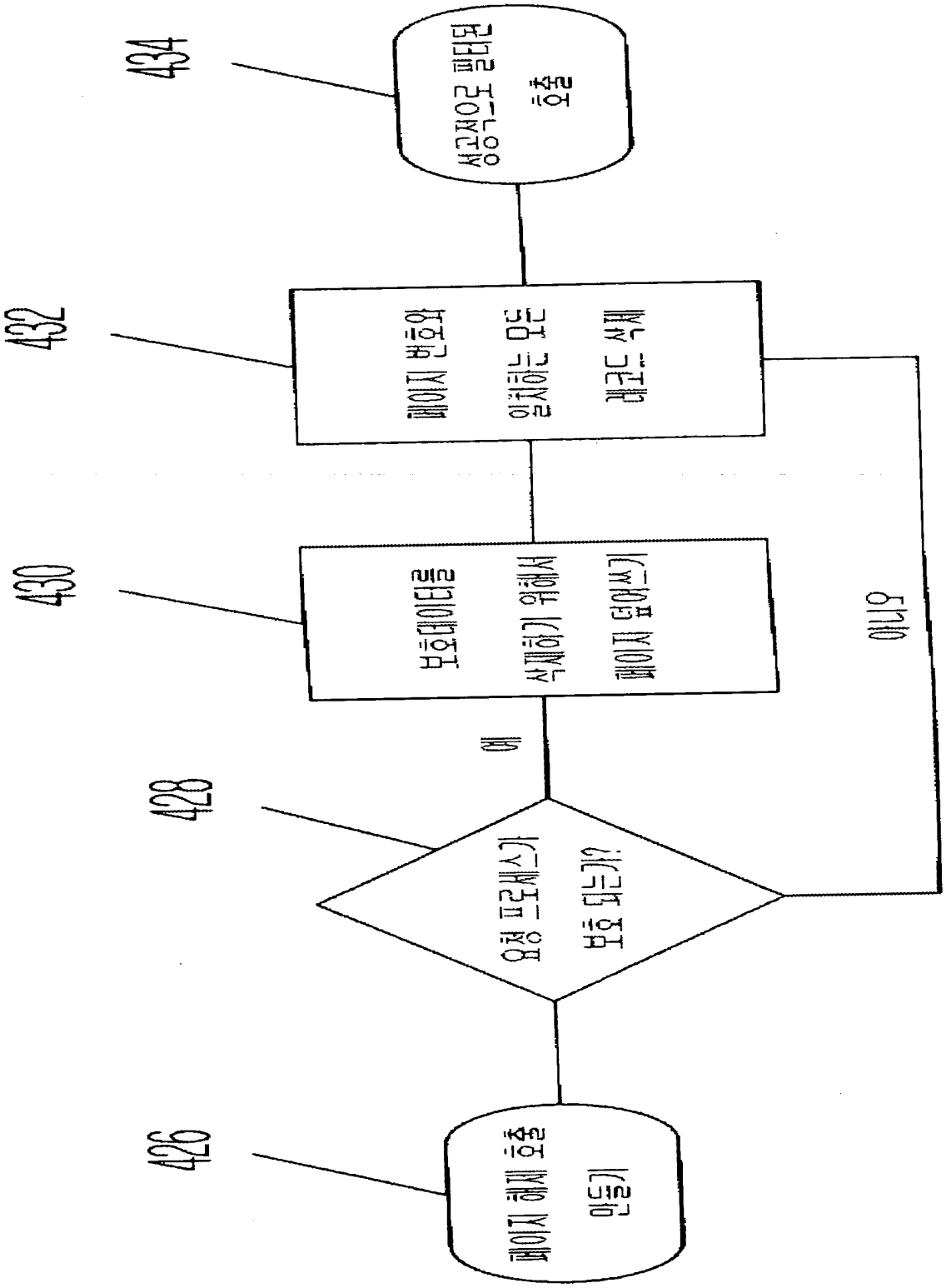
도면 5a


402
페이지 예약 호출 만들기
404
페이지는 공유될 수 있나? (요청의 속성 포함)
예
아니오
406
메모리는 공유될 수 없다. 트랙호출을 하지마라. 시스템이 요청을 완료하도록 허가
408
트랙 호출. 레코드는 프로세스 아이디, 페이지번호, 공유카운트(=0)로 생성
409
성공적으로 필터된 호출

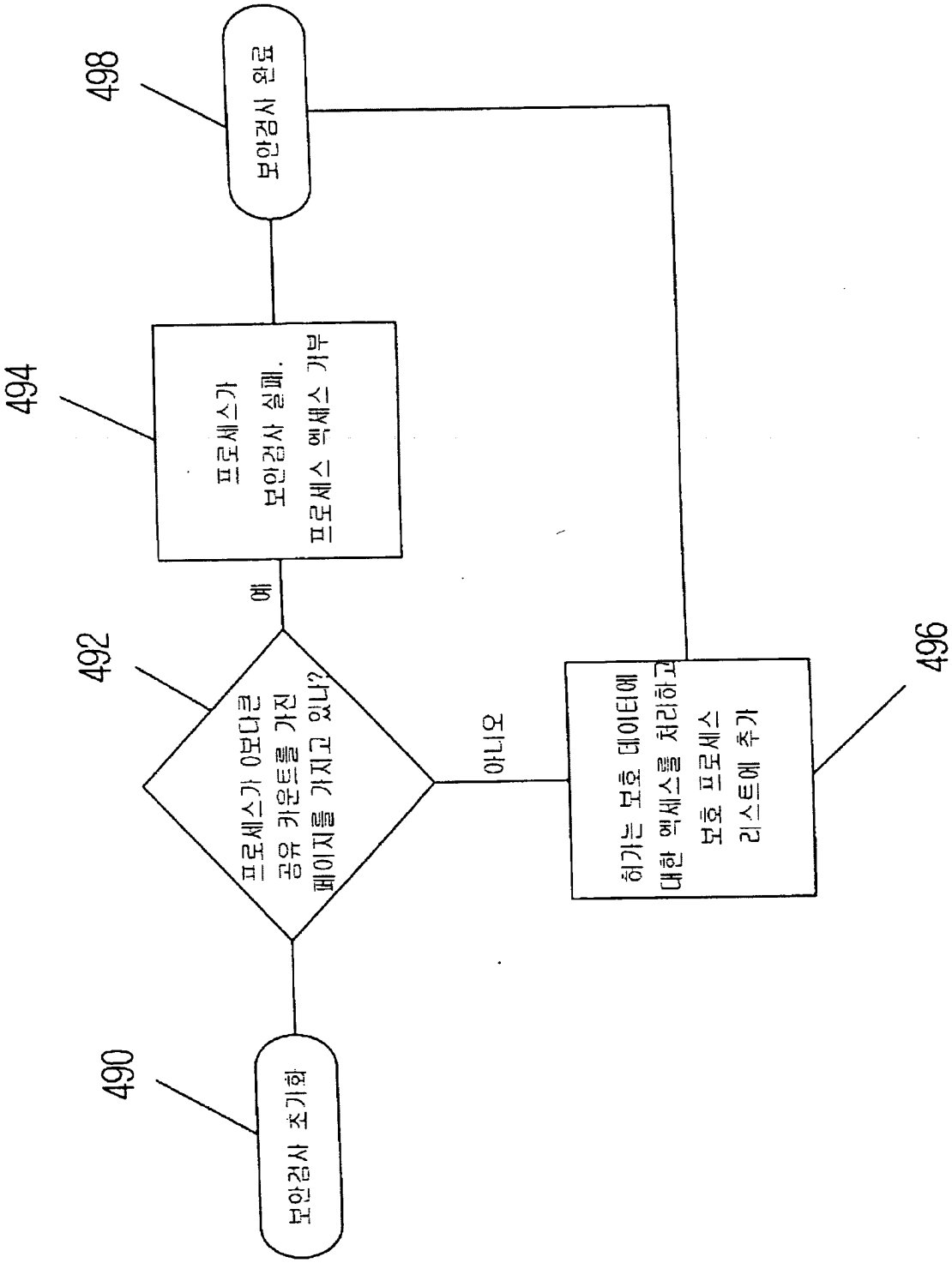
도면 5b


410
페이지 위임 호출 만들기
412
이 페이지는 다른 프로세스에 의해 공유됩니까?
예
414
이 프로세스 또는 공유 프로세스는 보호됩니까?
예
416
가절명령. 어떤 프로세스도 페이지가 보호 프로세스와 공유되도록 허가하지 않는다
418
성공적으로 밀터된 호출
아니오
424
이 요청은 호출된 프로세스에 의해 만들어 진다. 페이지 예약. 요청 완료
아니오
420
현재 프로세스 아이디, 페이지번호,공유카운트 (>0)와 함께 새 레코드 추가
322
새 공유카운트(+1)에서 같은 페이지 번호와 함께 이전에 존재하던 모든 레코드 업데이트

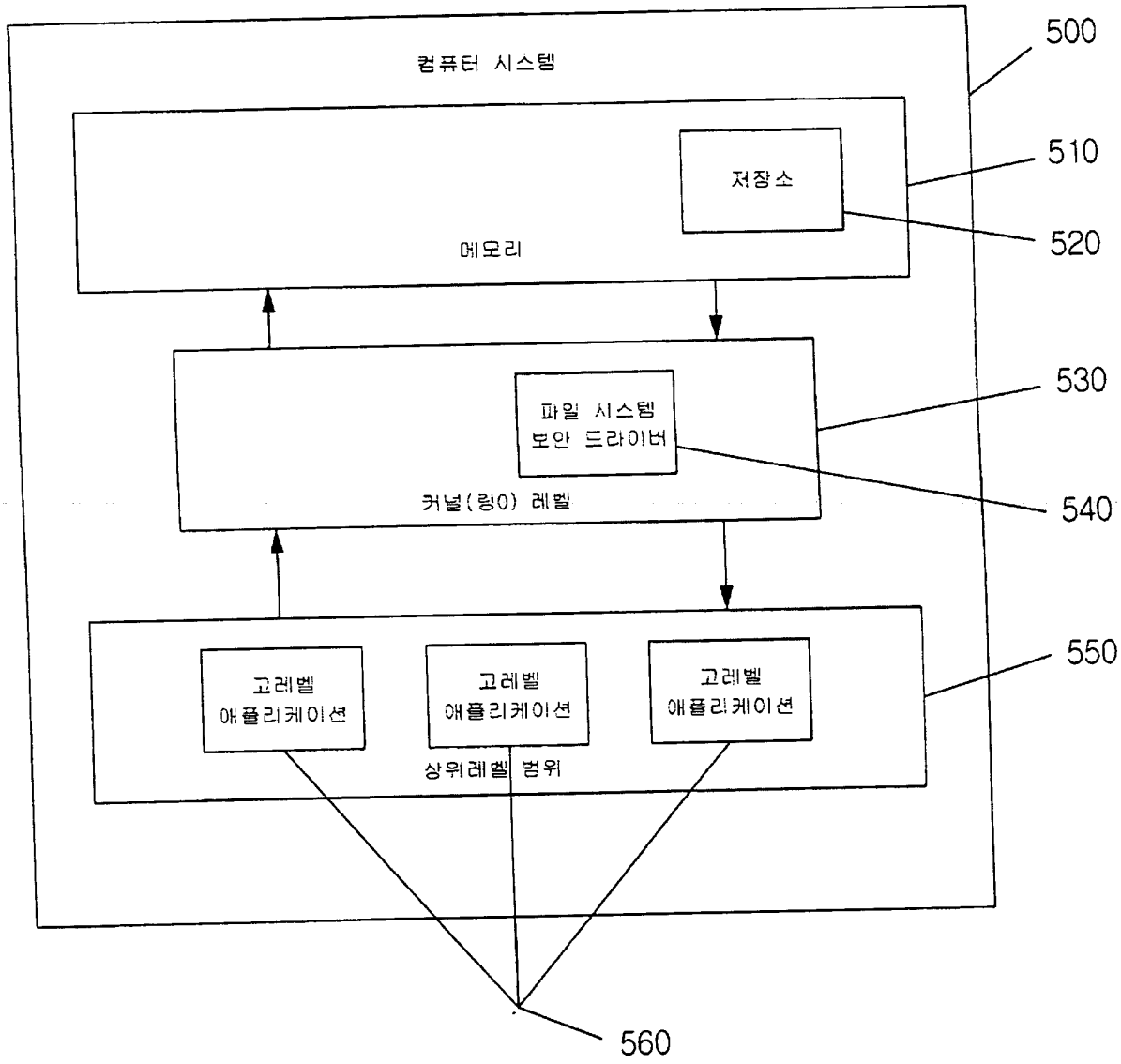
도면 5c


434
생고자으로 패턴된 호출
432
페이지 번호화 일치하는 모든 캐쉬드 삭제
430
보호데이터를 삭제하기 위해서 페이지 테이블스캔
아니오
예
428
요청 프로세스가 보호되는가?
426
페이지 해제 호출 만들기

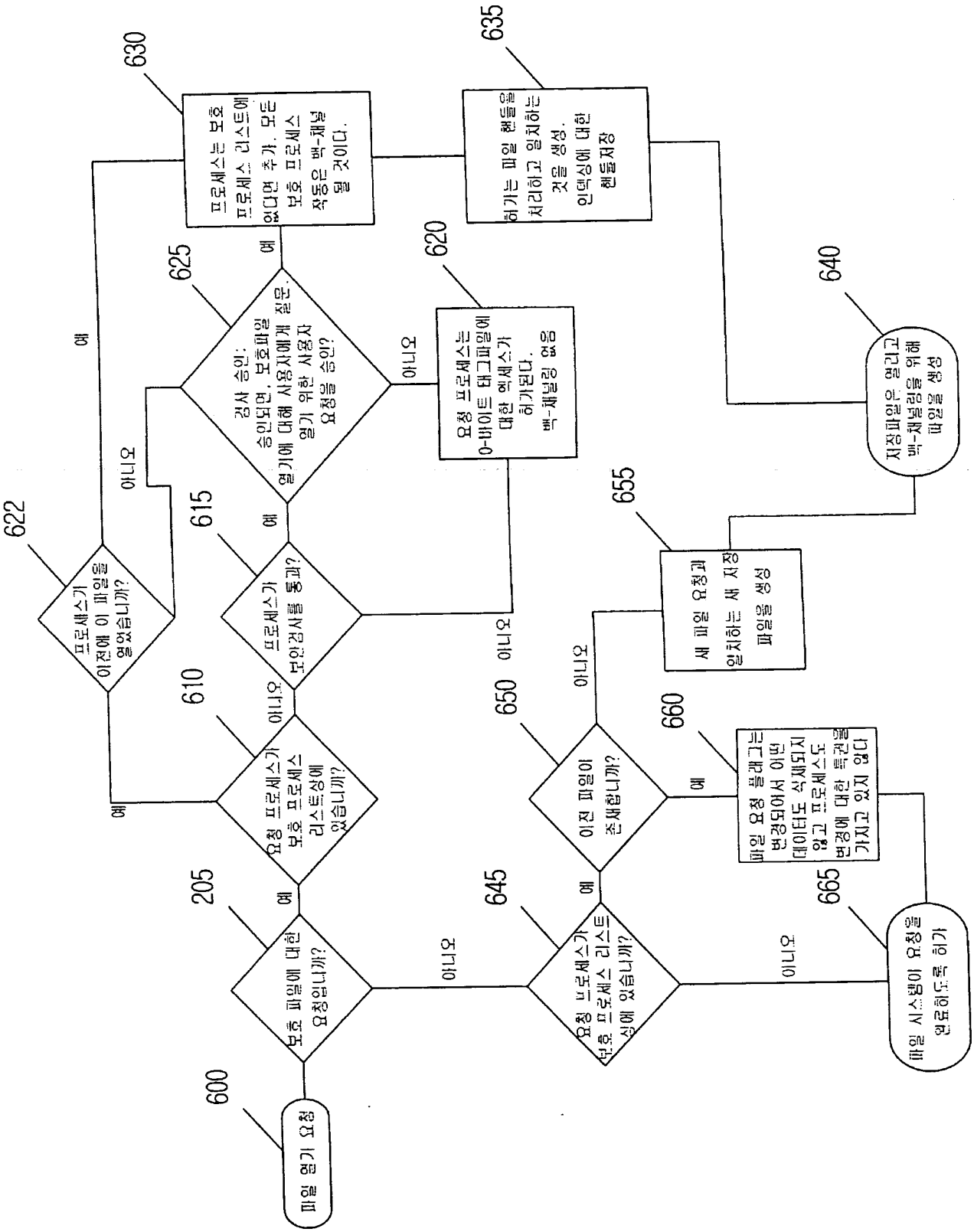
도면 5d


490
보안검사 초기화
492
프로세스가 0보다큰
공유 카운트를 가진
페이지를 가지고 있나?
예
494
프로세스가
보안검사 실패.
프로세스 엑세스 거부
498
보안검사 완료
아니오
496
허가는 보호 데이터에
대한 엑세스를 처리하고
보호 프로세스
리스트에 추가

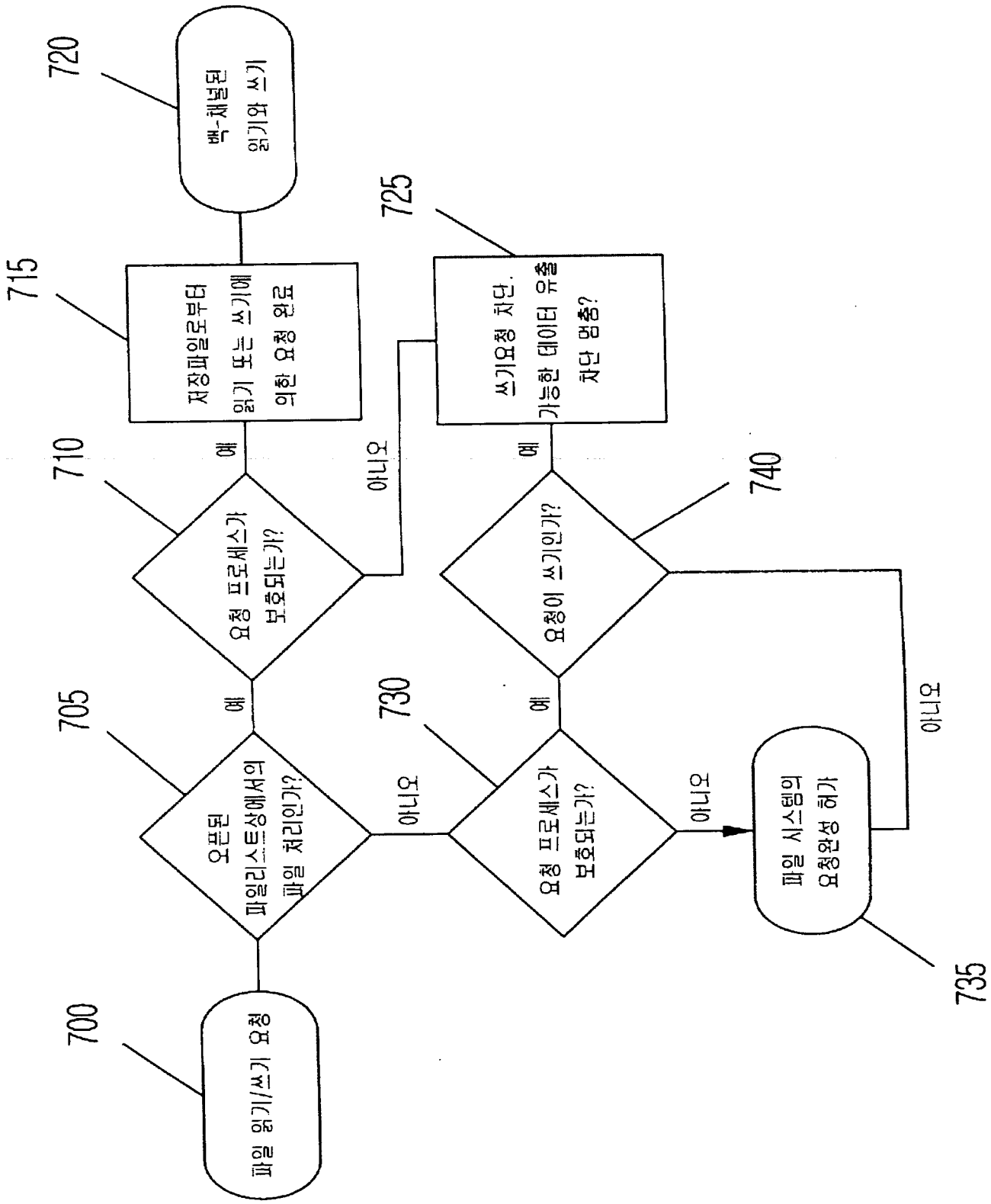
도면 6


500
컴퓨터 시스템
510
저장소
520
메모리
530
파일 시스템
보안 드라이버
540
커널(링0) 레벨
550
고레벨
애플리케이션
고레벨
애플리케이션
고레벨
애플리케이션
상위레벨 범위
560

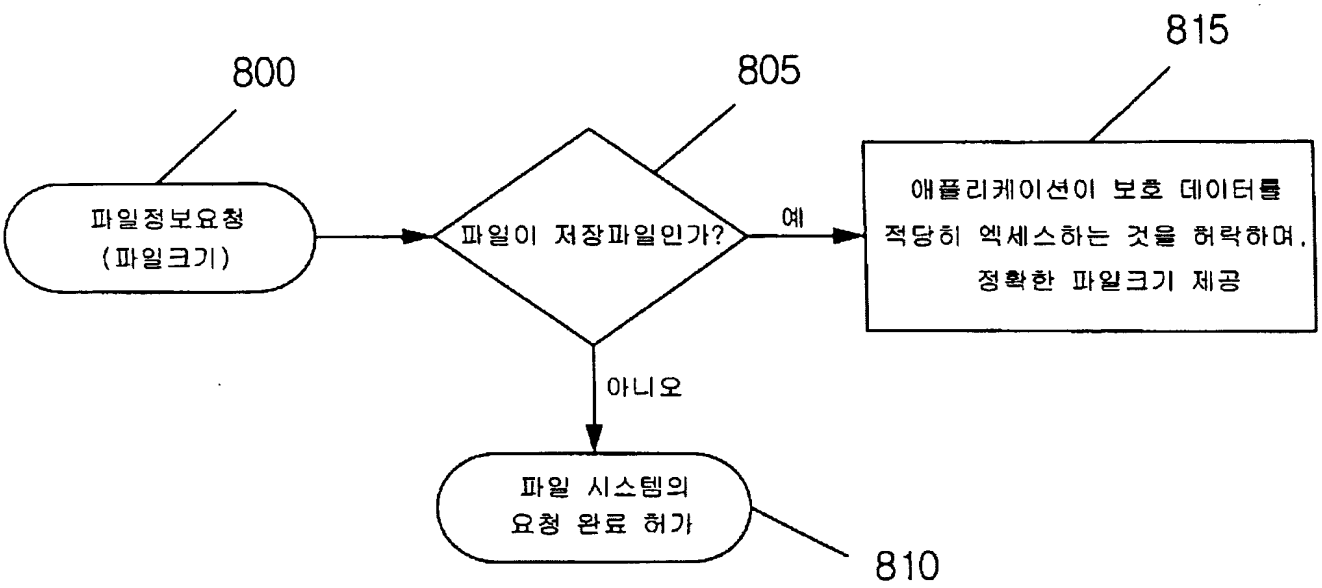
도면 7


600
파일 열기 요청
205
보호 파일에 대한 요청입니까?
예
아니오
610
요청 프로세스가 보호 프로세스 리스트상에 있습니까?
예
아니오
615
프로세스가 보안검사를 통과?
예
아니오
622
프로세스가 이전에 이 파일을 열었습니까?
예
아니오
625
검사 승인: 승인되면, 보호파일 열기에 대해 사용자에게 질문 요청을 승인?
예
아니오
630
프로세스는 보호 프로세스 리스트에 있다면 추가. 모든 보호 프로세스 작동은 백-채널 될 것이다.
620
요청 프로세스는 0-바이트 태그파일에 대한 엑세스가 허가된다. 백-채널링 없음
635
허가는 파일 핸들을 처리하고 일치하는 것을 생성. 인덱싱에 대한 핸들저장
640
저장파일은 열리고 백-채널링을 위해 파일을 생성
645
요청 프로세스가 보호 프로세스 리스트 상에 있습니까?
예
아니오
650
이전 파일이 존재합니까?
예
아니오
655
새 파일 요청과 일치하는 새 저장 파일을 생성
660
파일 요청 플래그는 변경되어서 이전 데이터도 삭제되지 않고 프로세스도 변경에 대한 특권을 가지고 있지 않다
665
파일 시스템이 요청을 완료하도록 허가

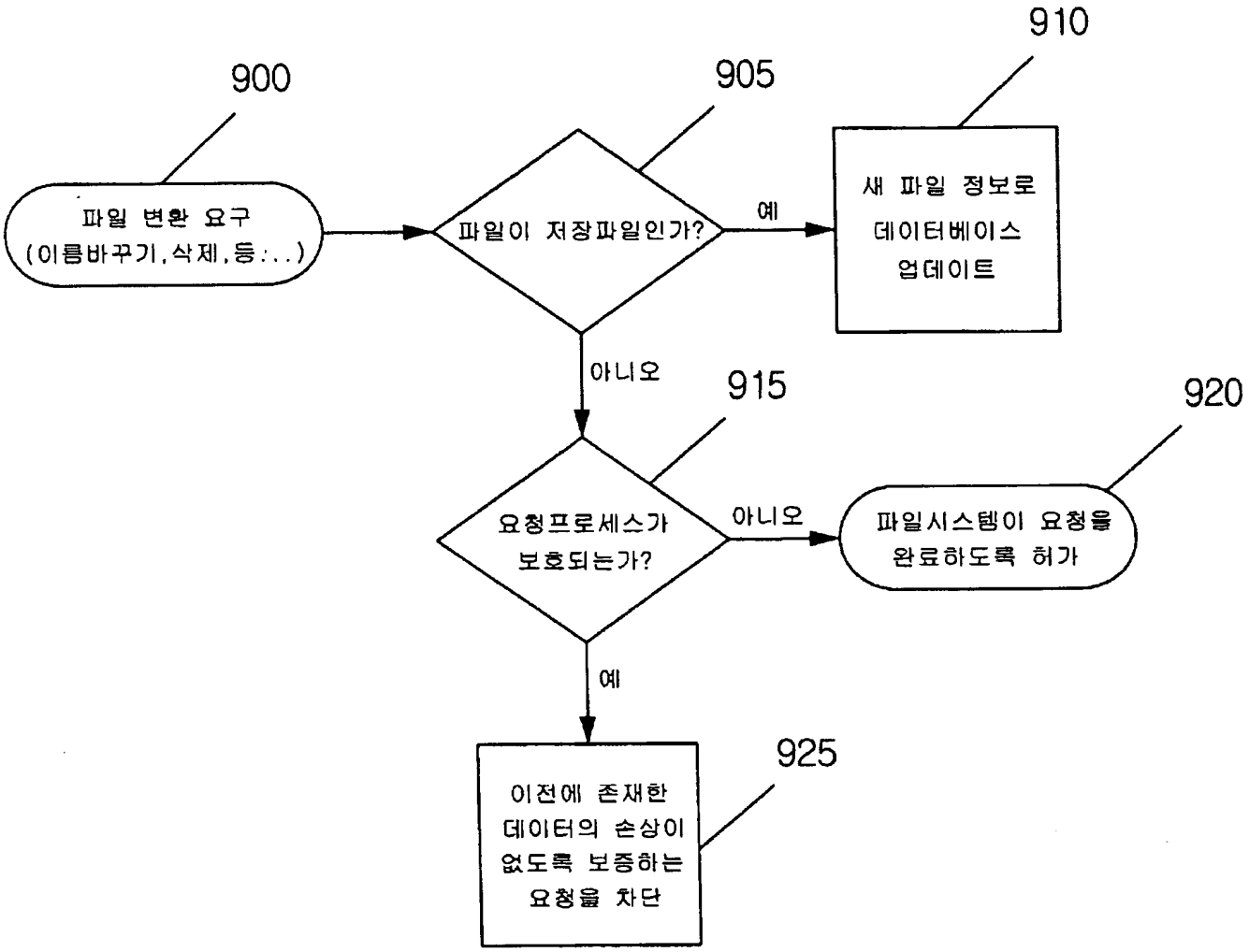
도면 8


700
파일 읽기/쓰기 요청
705
오픈된 파일리스트상에서의 파일 처리인가?
예
아니오
710
요청 프로세스가 보호되는가?
예
아니오
715
저장파일로부터 읽기 또는 쓰기에 의한 요청 완료
720
백-채널된 읽기와 쓰기
725
쓰기요청 차단. 가능한 데이터 유출 차단 멈춤?
730
요청 프로세스가 보호되는가?
예
아니오
735
파일 시스템의 요청완성 허가
740
요청이 쓰기인가?
예
아니오

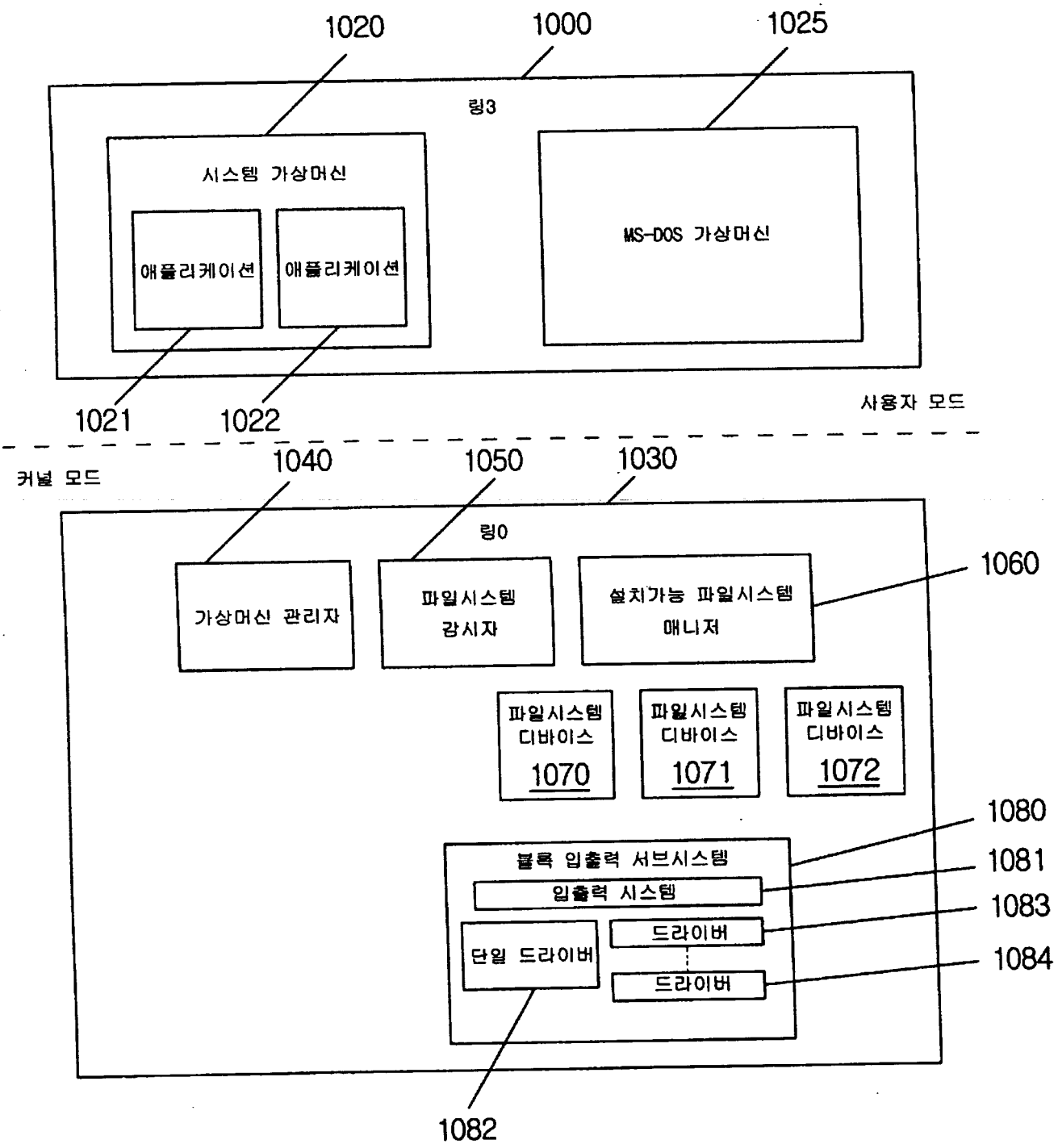
도면 9


800
805
815
파일정보요청
(파일크기)
파일이 저장파일인가?
예
애플리케이션이 보호 데이터를
적당히 엑세스하는 것을 허락하며,
정확한 파일크기 제공
아니오
파일 시스템의
요청 완료 허가
810

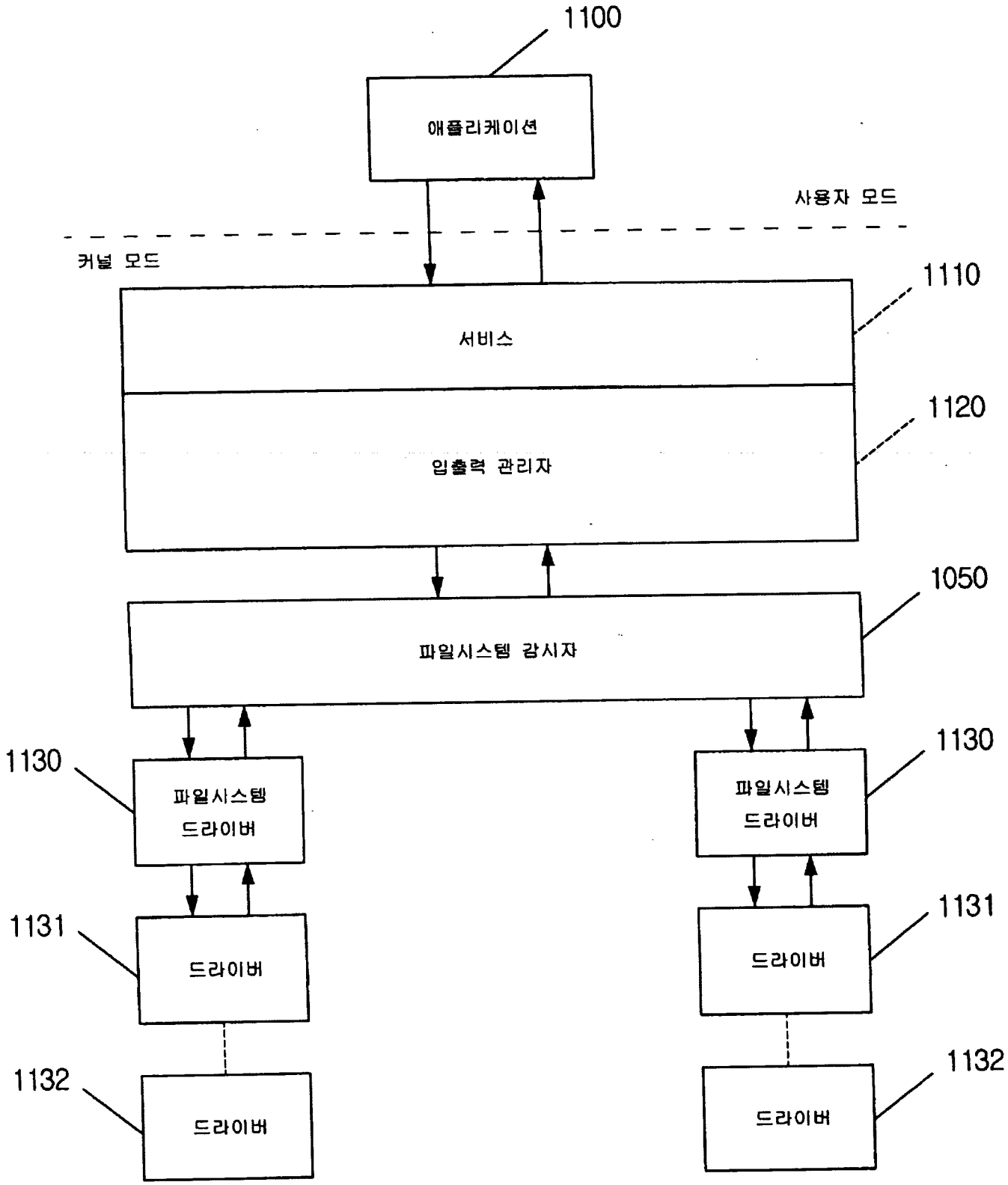
도면 10


900
파일 변환 요구
(이름바꾸기,삭제,등:..)
905
파일이 저장파일인가?
예
910
새 파일 정보로
데이터베이스
업데이트
아니오
915
요청프로세스가
보호되는가?
아니오
920
파일시스템이 요청을
완료하도록 허가
예
925
이전에 존재한
데이터의 손상이
없도록 보증하는
요청을 차단

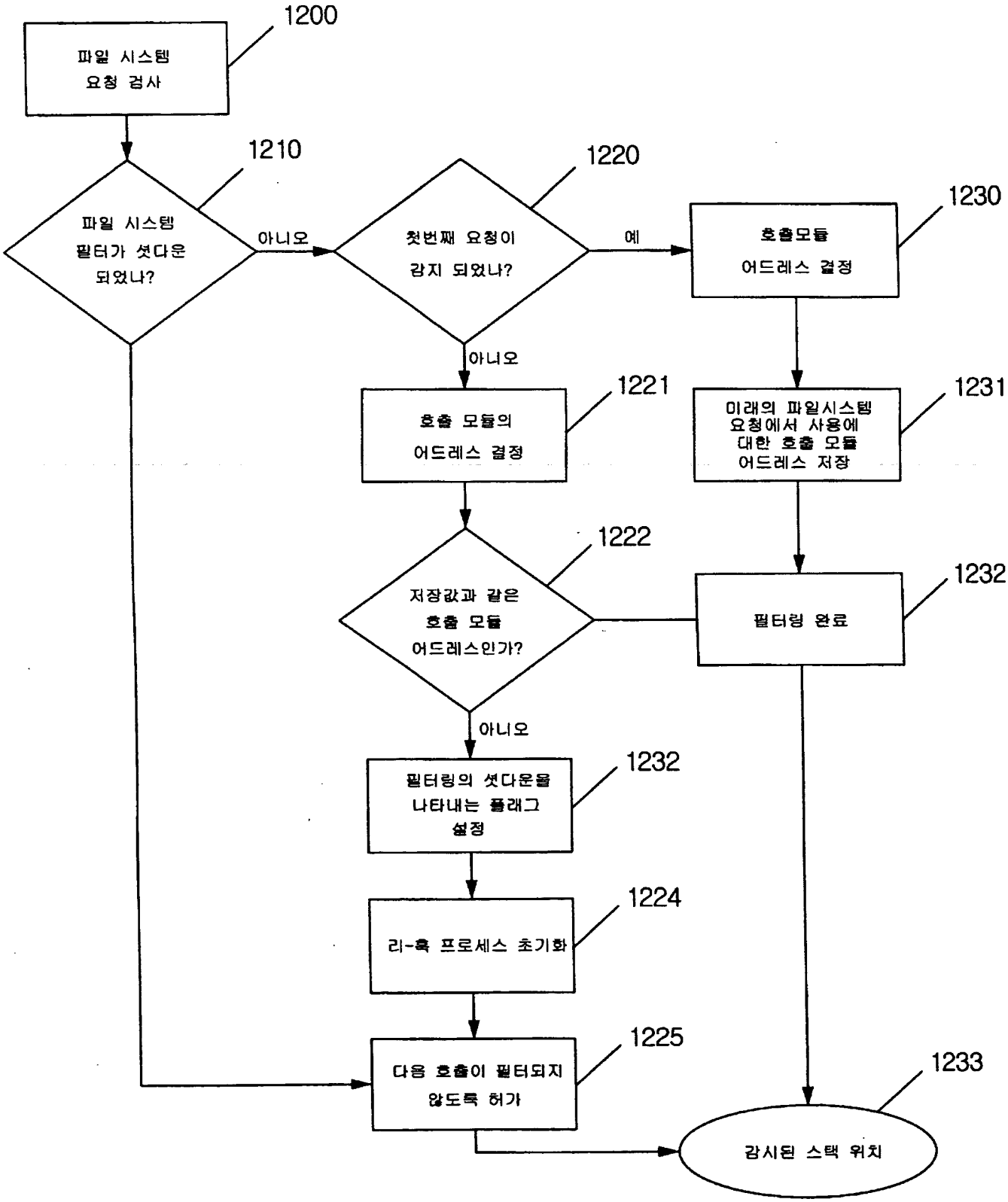
도면 11


1020
1000
1025
링3
시스템 가상머신
애플리케이션
애플리케이션
MS-DOS 가상머신
1021
1022
사용자 모드
커널 모드
1040
1050
1030
링0
가상머신 관리자
파일시스템 감시자
설치가능 파일시스템 매니저
1060
파일시스템 디바이스 1070
파일시스템 디바이스 1071
파일시스템 디바이스 1072
1080
블록 입출력 서브시스템
1081
입출력 시스템
1083
단일 드라이버
드라이버
1084
드라이버
1082

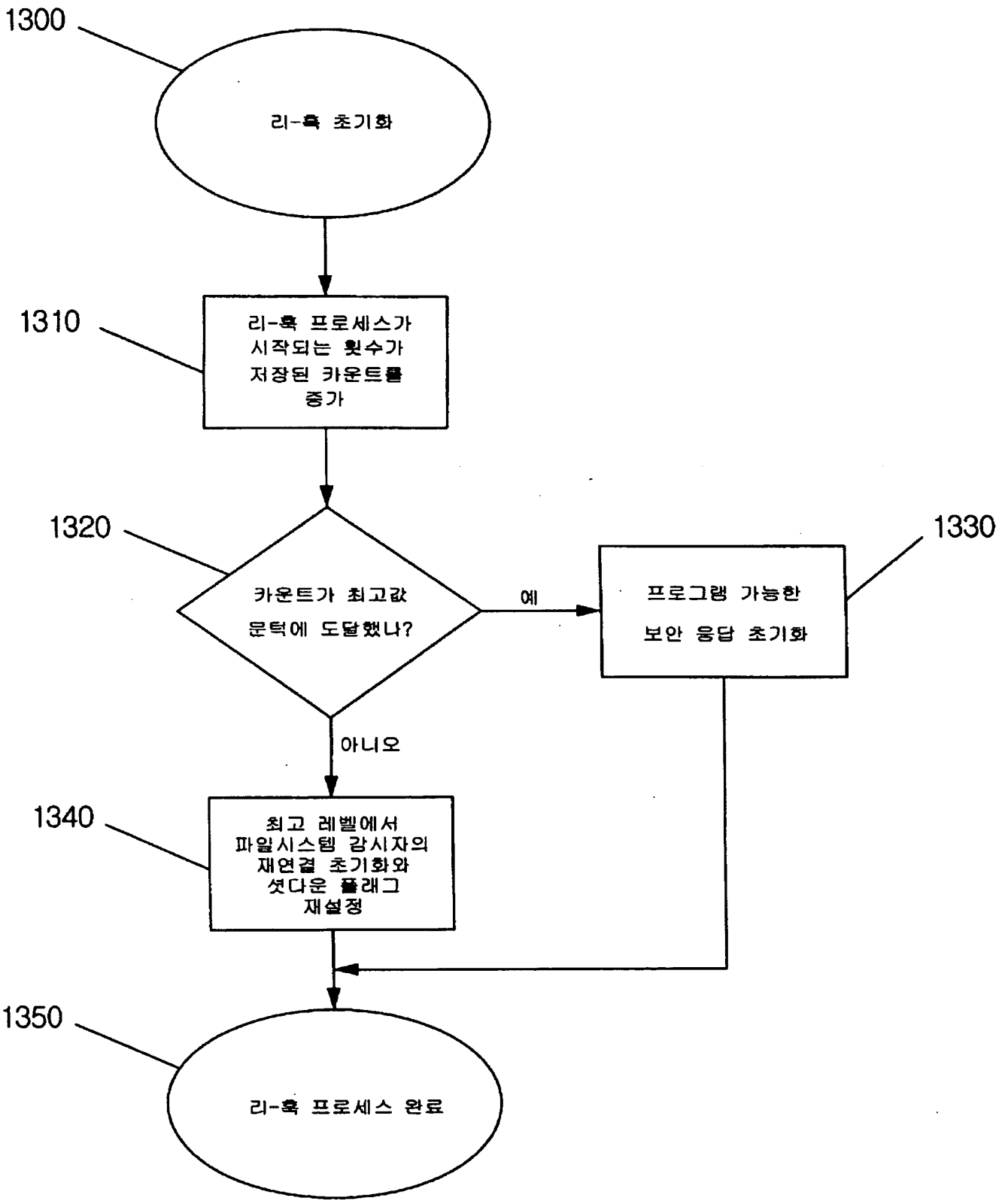
도면 12


1100
애플리케이션
사용자 모드
커널 모드
1110
서비스
1120
입출력 관리자
1050
파일시스템 감시자
1130
파일시스템
드라이버
1130
파일시스템
드라이버
1131
드라이버
1131
드라이버
1132
드라이버
1132
드라이버

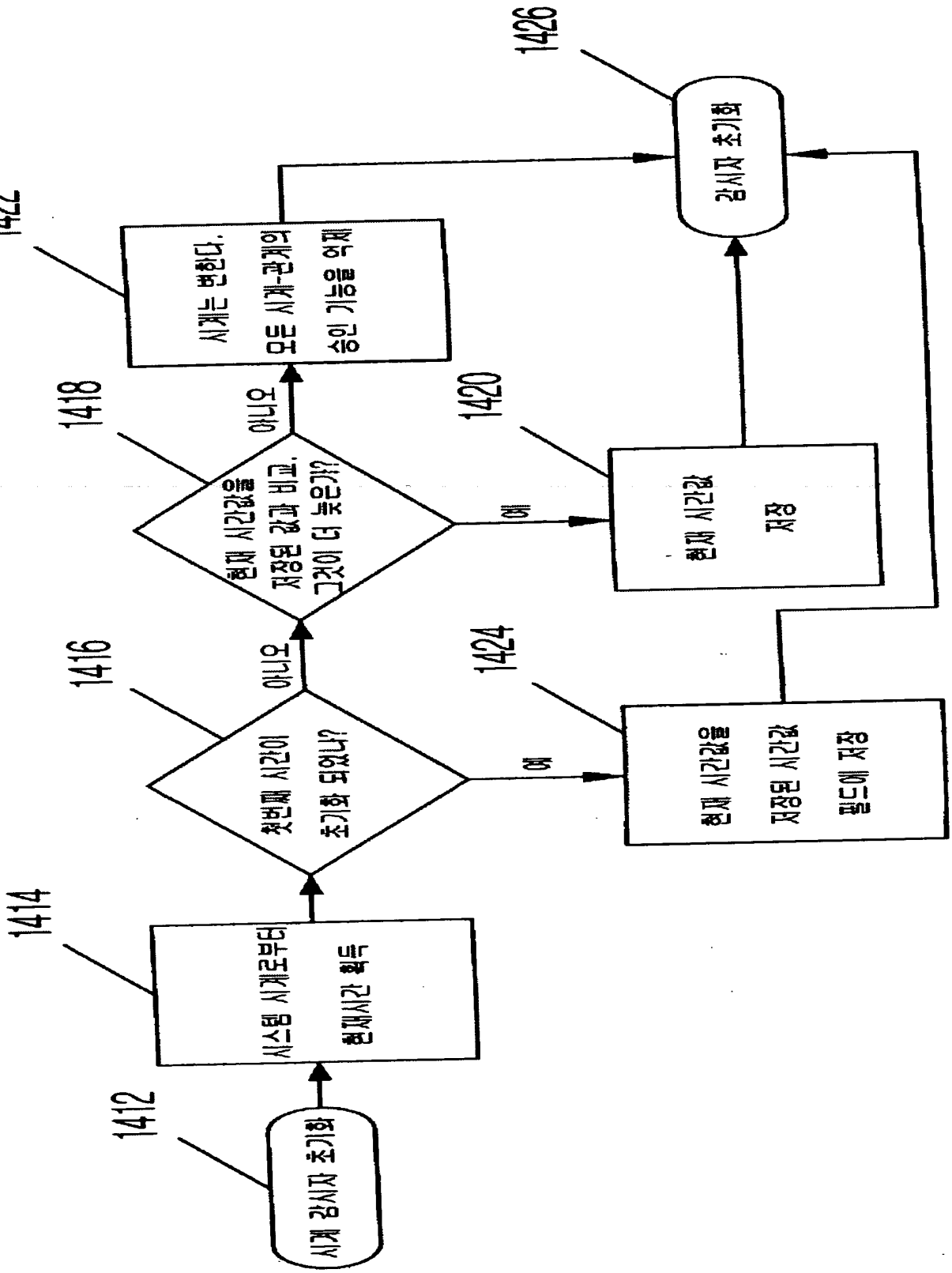
도면 13


1200
파일 시스템
요청 검사
1210
파일 시스템
필터가 셧다운
되었나?
아니오
1220
첫번째 요청이
감지 되었나?
예
1230
호출모듈
어드레스 결정
아니오
1221
호출 모듈의
어드레스 결정
1231
미래의 파일시스템
요청에서 사용에
대한 호출 모듈
어드레스 저장
1222
저장값과 같은
호출 모듈
어드레스인가?
1232
필터링 완료
아니오
1232
필터링의 셧다운을
나타내는 플래그
설정
1224
리-엑 프로세스 초기화
1225
다음 호출이 필터되지
않도록 허가
1233
감시된 스택 위치

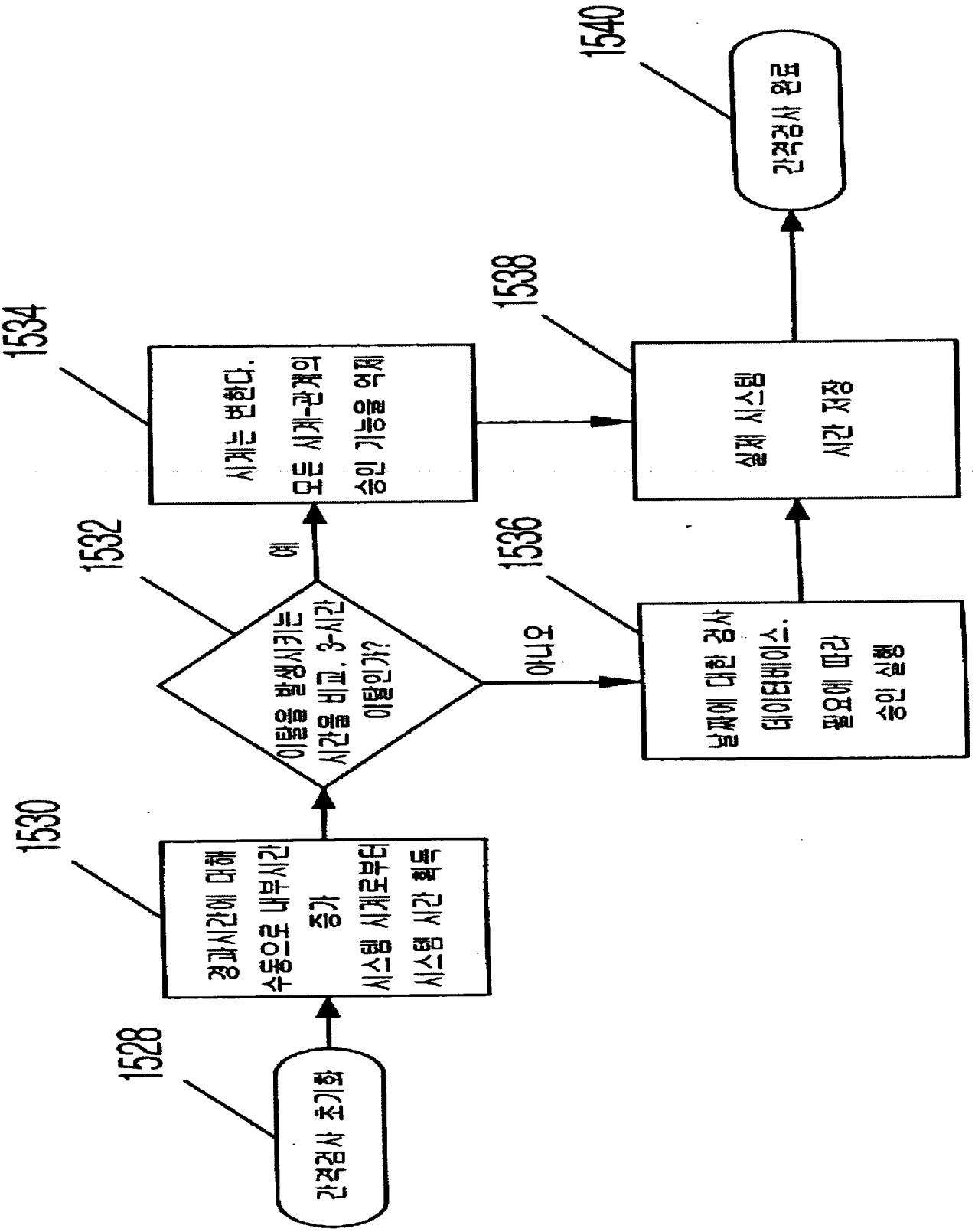
도면 14


1300
리-훅 초기화
1310
리-훅 프로세스가 시작되는 횟수가 저장된 카운트를 증가
1320
카운트가 최고값 문턱에 도달했나?
예
1330
프로그램 가능한 보안 응답 초기화
아니오
1340
최고 레벨에서 파일시스템 감시자의 재연결 초기화와 셧다운 플래그 재설정
1350
리-훅 프로세스 완료

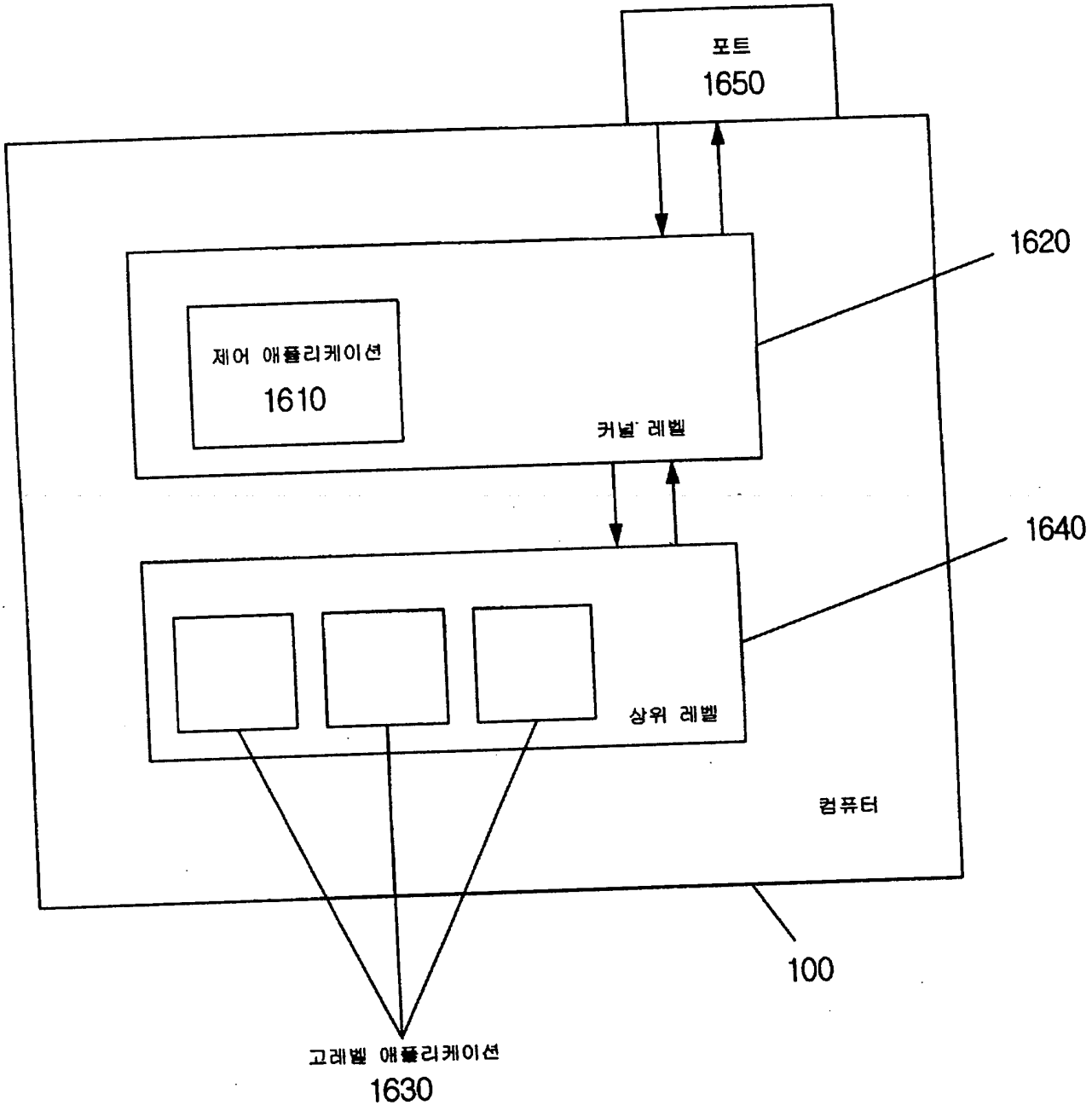
도면 15


1412
시계 감시자 초기화
1414
시스템 시계로부터 현재시간 획득
1416
첫번째 시간이 초기화 되었나?
아니오
예
1418
현재 시간값을 저장된 값과 비교, 그것이 더 늦은가?
아니오
예
1420
현재 시간값 저장
1422
시계는 변한다. 모든 시계-관계의 승인 기능을 억제
1424
현재 시간값을 저장된 시간값 필드에 저장
1426
감시자 초기화

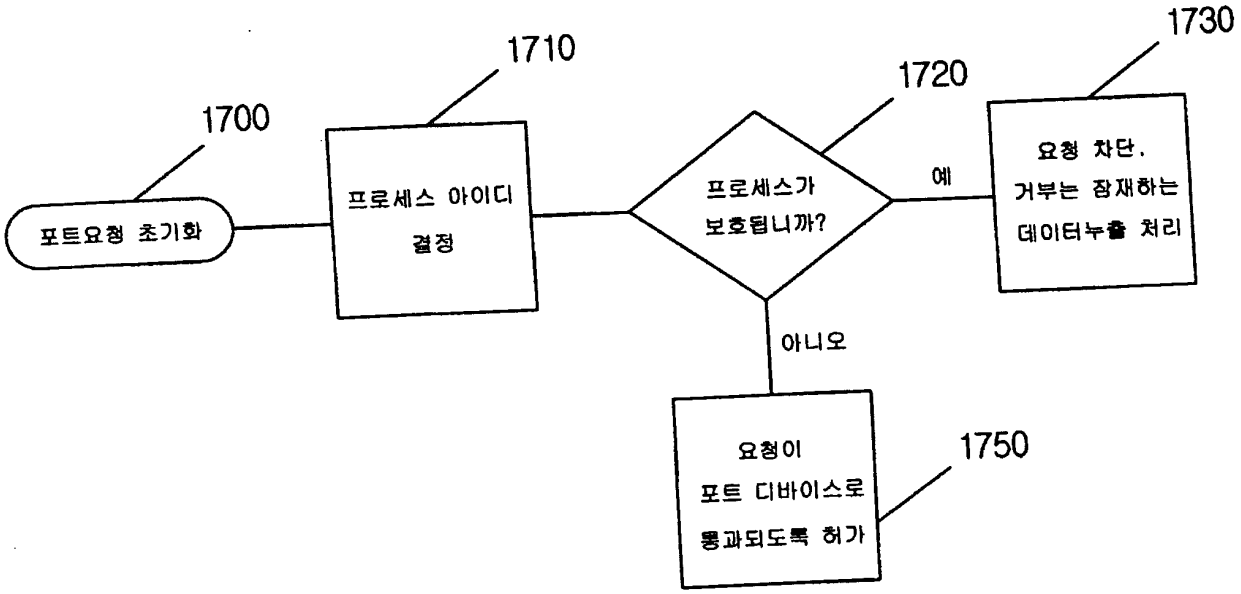
도면 16


1528
간격검사 초기화
1530
정교시간에 대해 수동으로 내부시간 칩 시스템 시계로부터 시스템 시간 획득
1532
이들을 발생시키는 시간을 비교. 3-시간 이달인가?
예
아니오
1534
시계는 변한다. 모든 시계-관계의 승인 기능을 억제
1536
남겨에 대한 검사 데이터베이스. 필요에 따라 승인 실행
1538
실제 시스템 시간 저장
1540
간격검사 완료

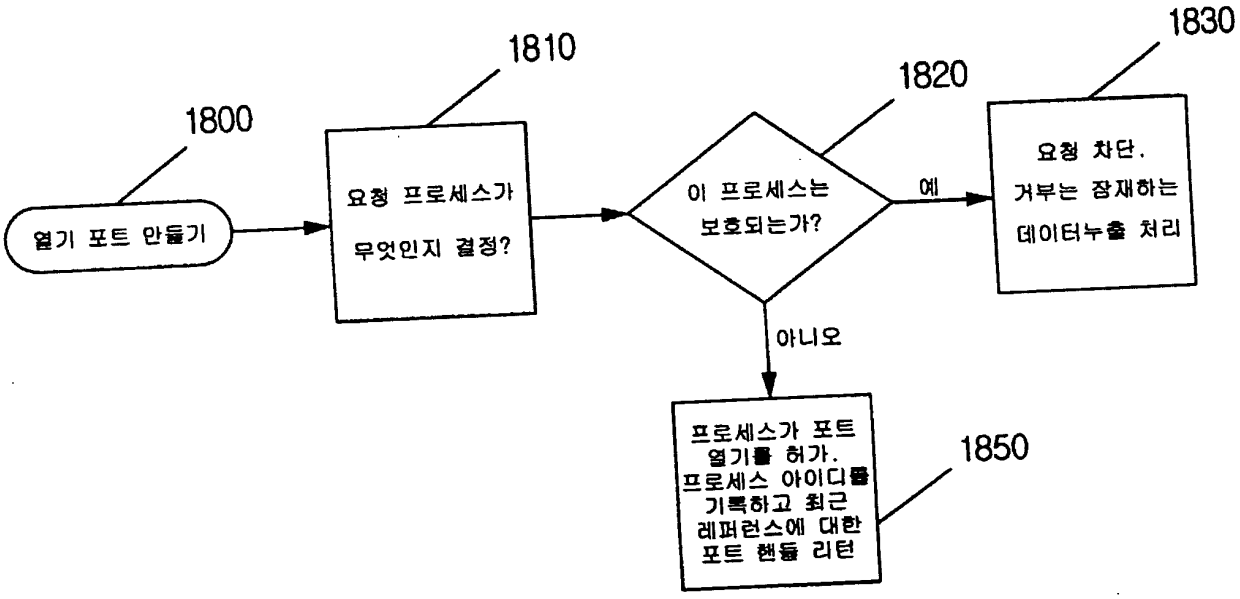
도면 17


포트
1650
1620
제어 애플리케이션
1610
커널 레벨
1640
상위 레벨
컴퓨터
100
고레벨 애플리케이션
1630

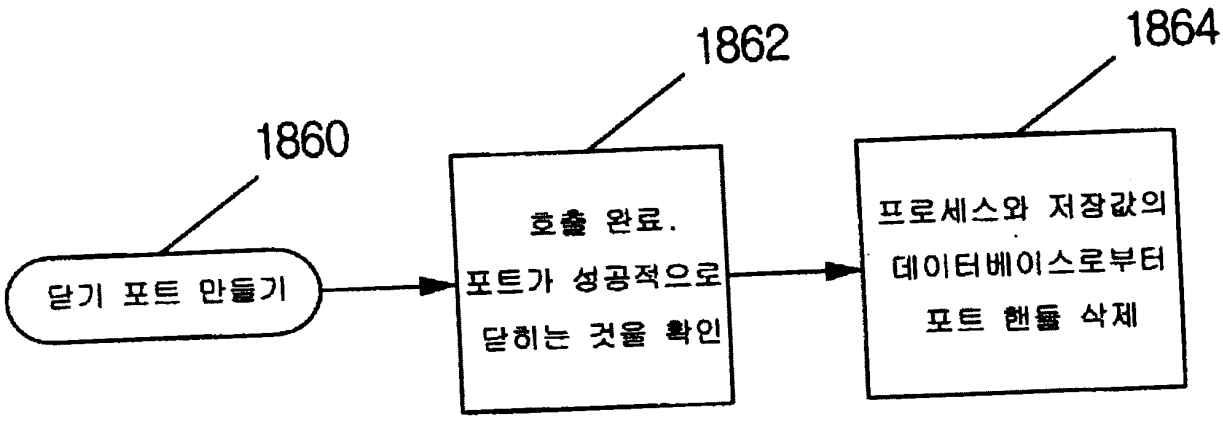
도면 18


1700
포트요청 초기화
1710
프로세스 아이디 결정
1720
프로세스가 보호됩니까?
예
1730
요청 차단. 거부는 장재하는 데이터누출 처리
아니오
요청이 포트 디바이스로 통과되도록 허가
1750

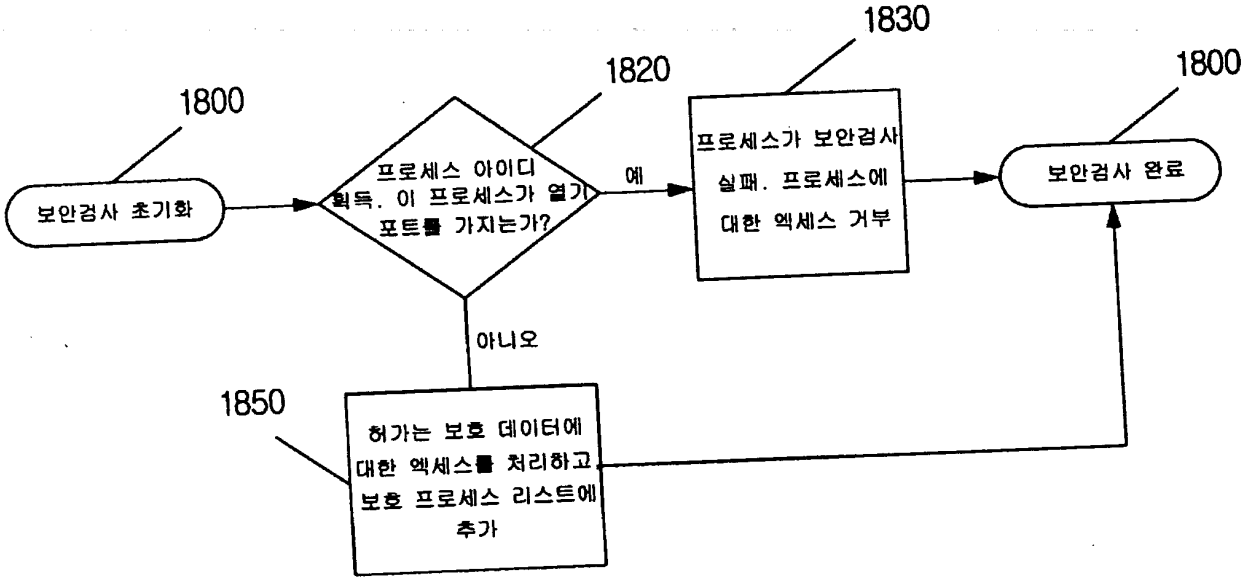
도면 19a


1800
열기 포트 만들기
1810
요청 프로세스가 무엇인지 결정?
1820
이 프로세스는 보호되는가?
예
1830
요청 차단. 거부는 장재하는 데이터누출 처리
아니오
프로세스가 포트 열기를 허가. 프로세스 아이디를 기록하고 최근 레퍼런스에 대한 포트 핸들 리턴
1850

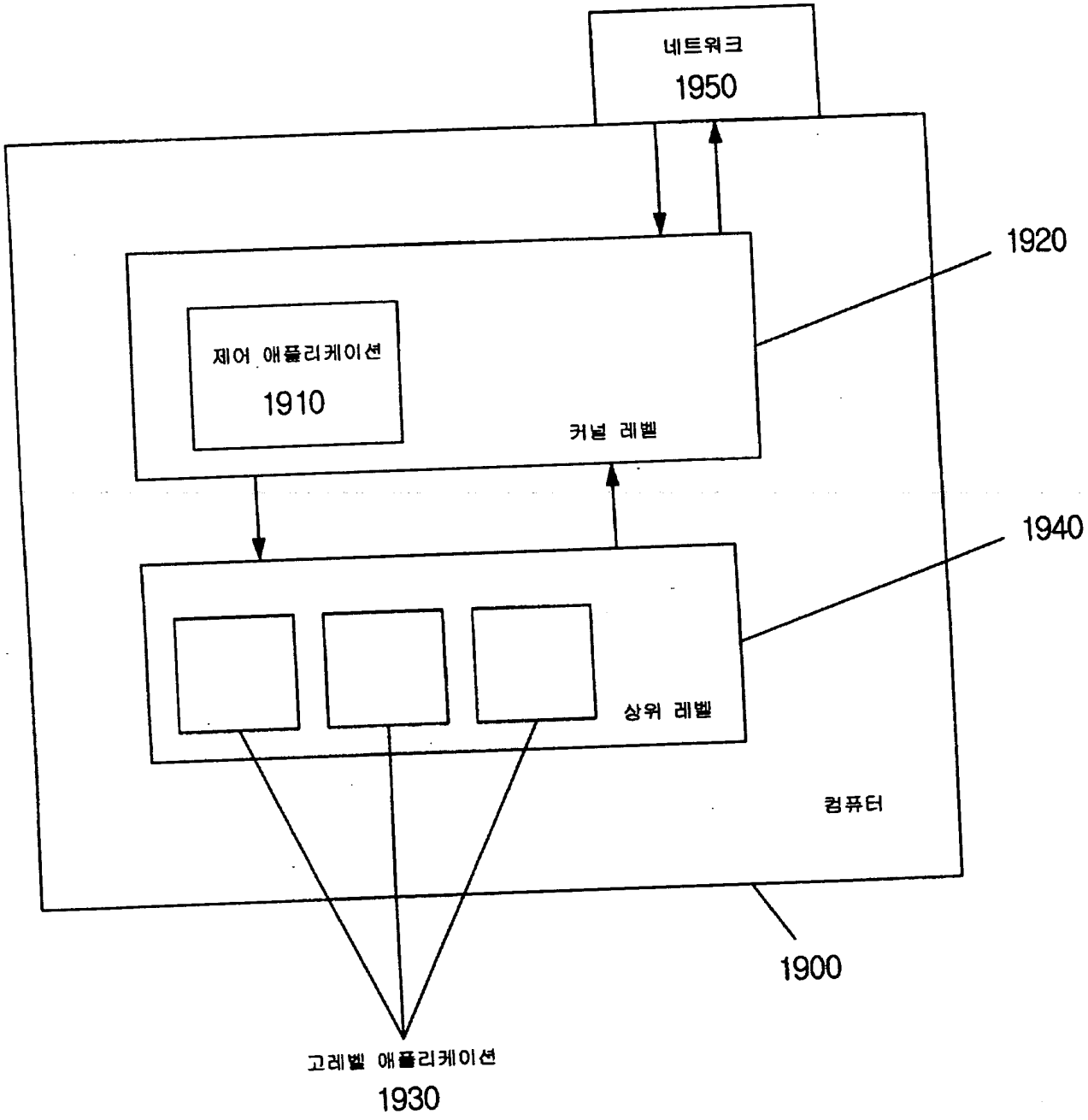
도면 19b


1860
닫기 포트 만들기
1862
호출 완료.
포트가 성공적으로
닫히는 것을 확인
1864
프로세스와 저장값의
데이터베이스로부터
포트 핸들 삭제

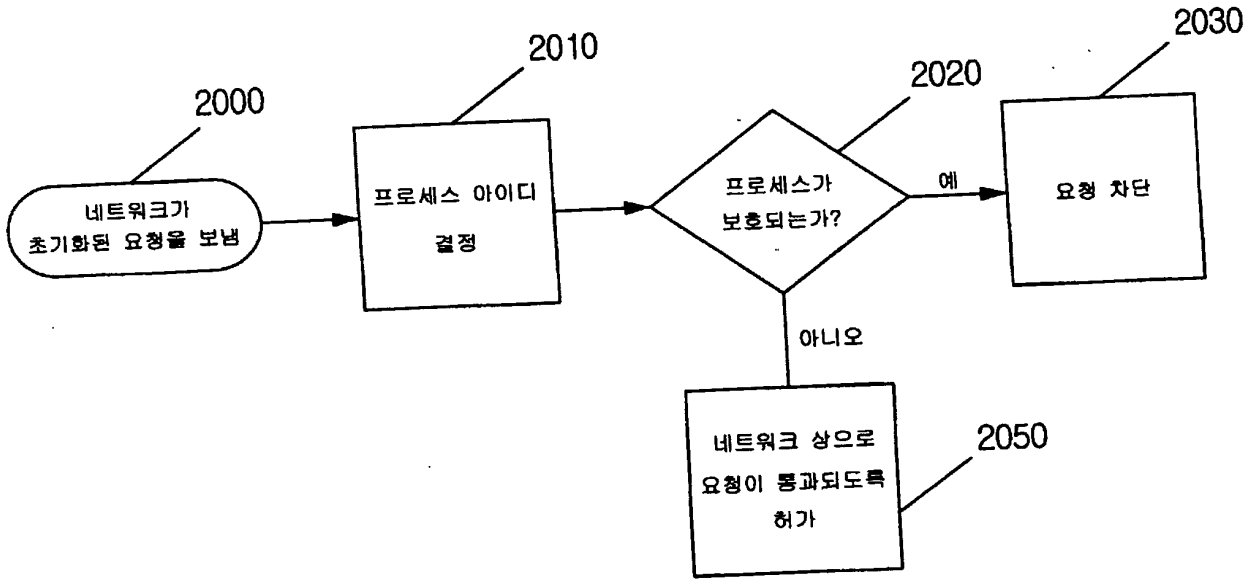
도면 19c


1800
보안검사 초기화
1820
프로세스 아이디
획득. 이 프로세스가 열기
포트를 가지는가?
예
1830
프로세스가 보안검사
실패. 프로세스에
대한 엑세스 거부
1800
보안검사 완료
아니오
1850
허가는 보호 데이터에
대한 엑세스를 처리하고
보호 프로세스 리스트에
추가

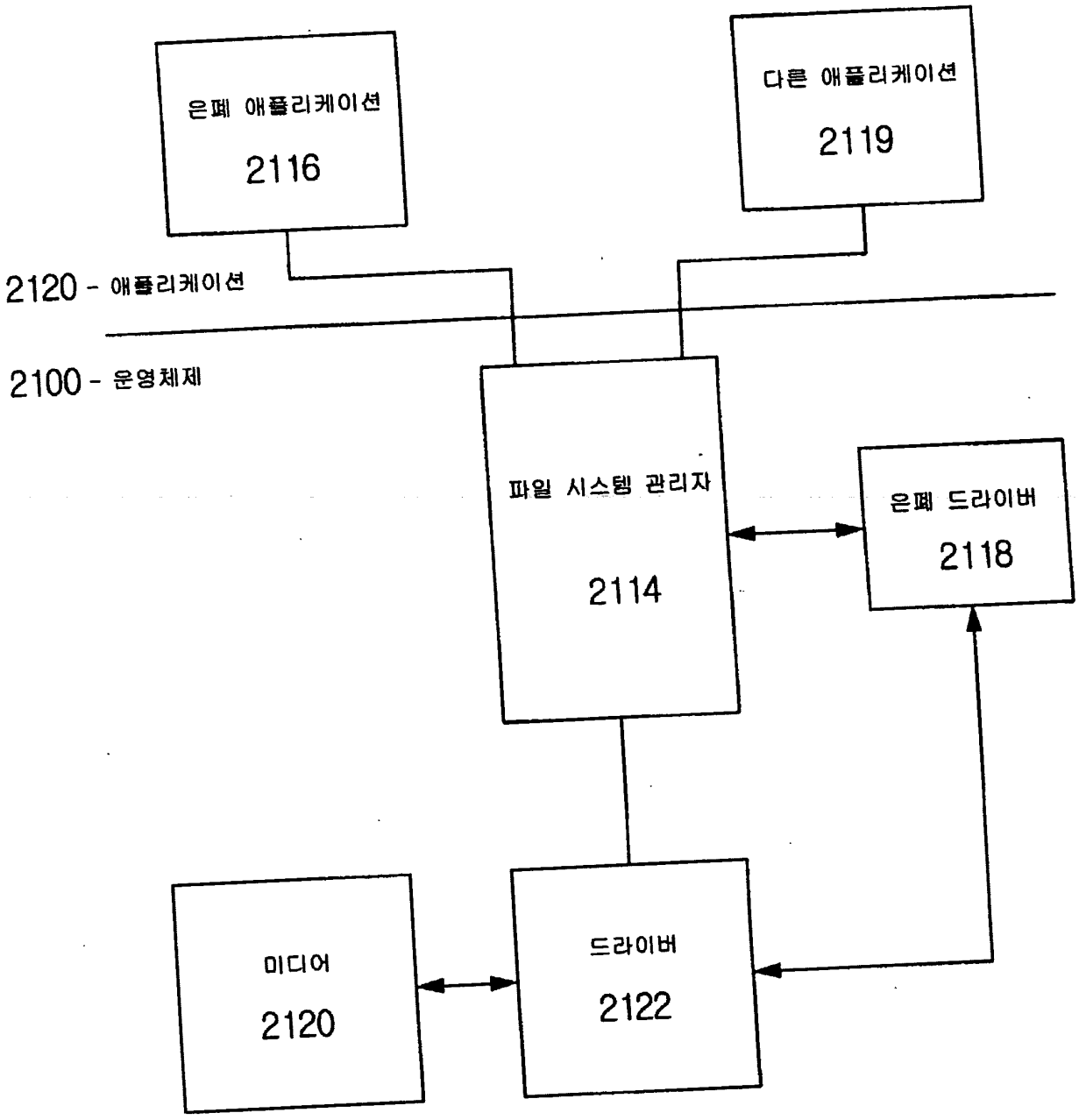
도면 20


네트워크
1950
1920
제어 애플리케이션
1910
커널 레벨
1940
상위 레벨
컴퓨터
1900
고레벨 애플리케이션
1930

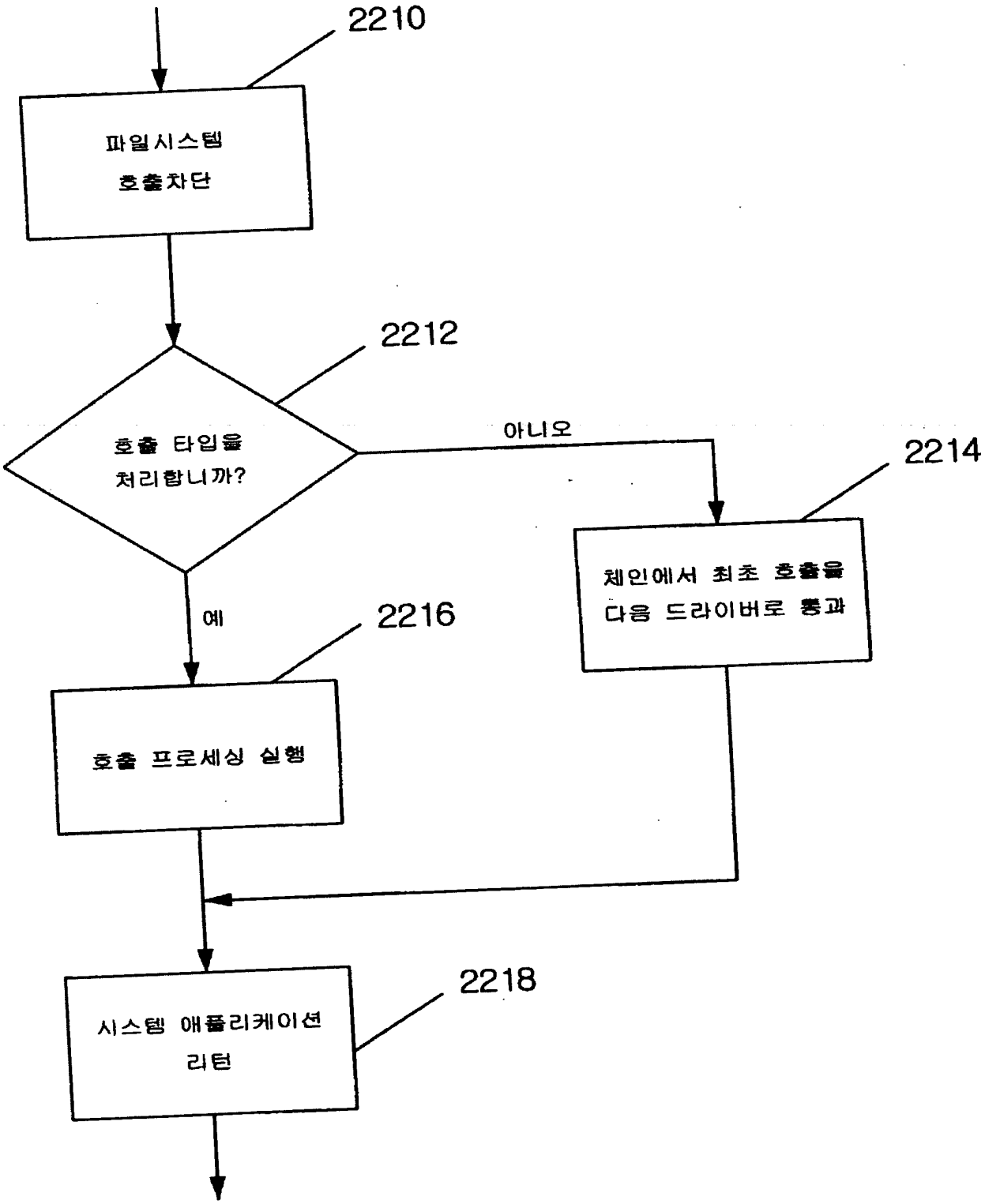
도면 21


2000
네트워크가
초기화된 요청을 보냄
2010
프로세스 아이디
결정
2020
프로세스가
보호되는가?
예
2030
요청 차단
아니오
2050
네트워크 상으로
요청이 통과되도록
허가

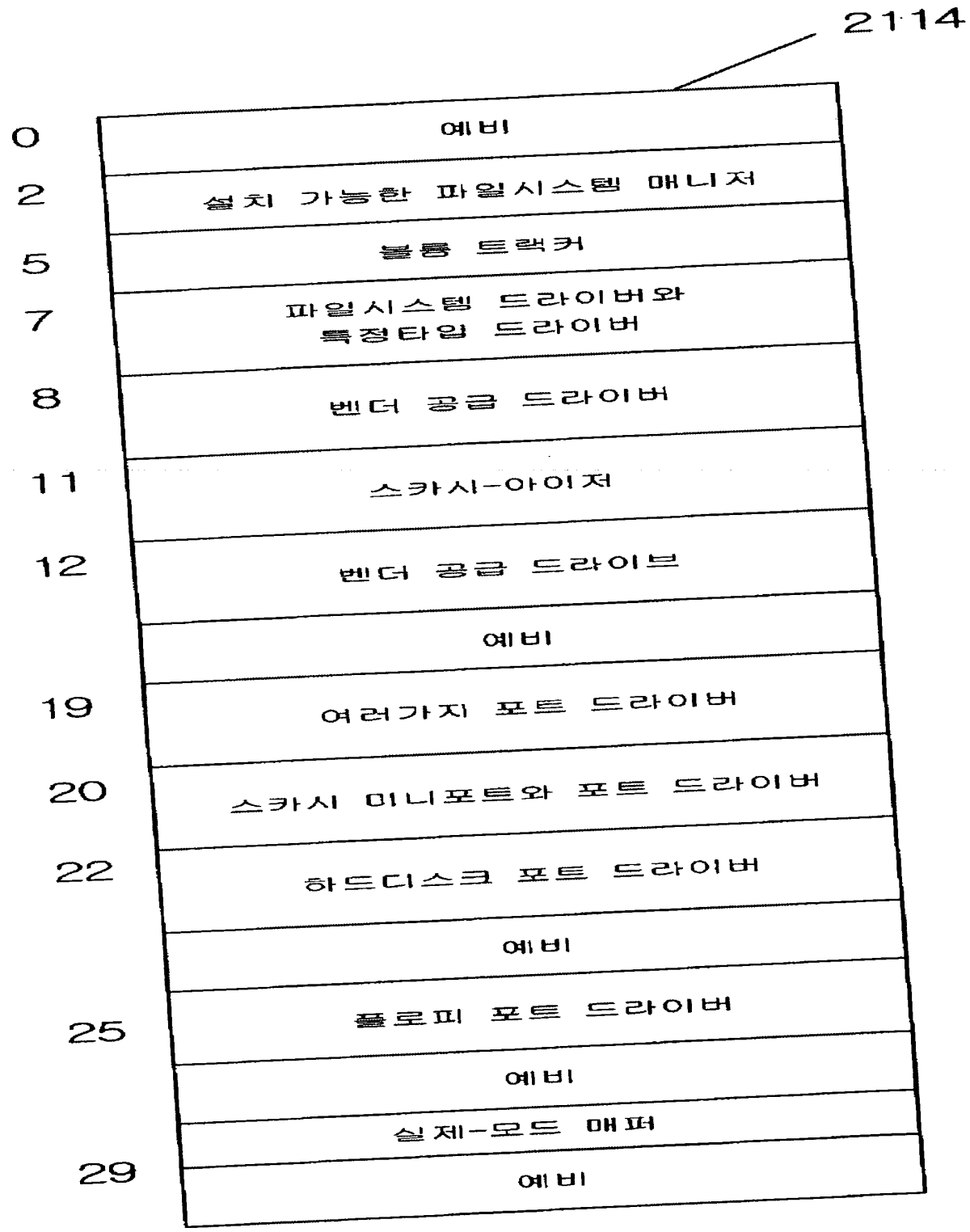
도면 22


은폐 애플리케이션
2116
다른 애플리케이션
2119
2120 - 애플리케이션
2100 - 운영체제
파일 시스템 관리자
2114
은폐 드라이버
2118
미디어
2120
드라이버
2122

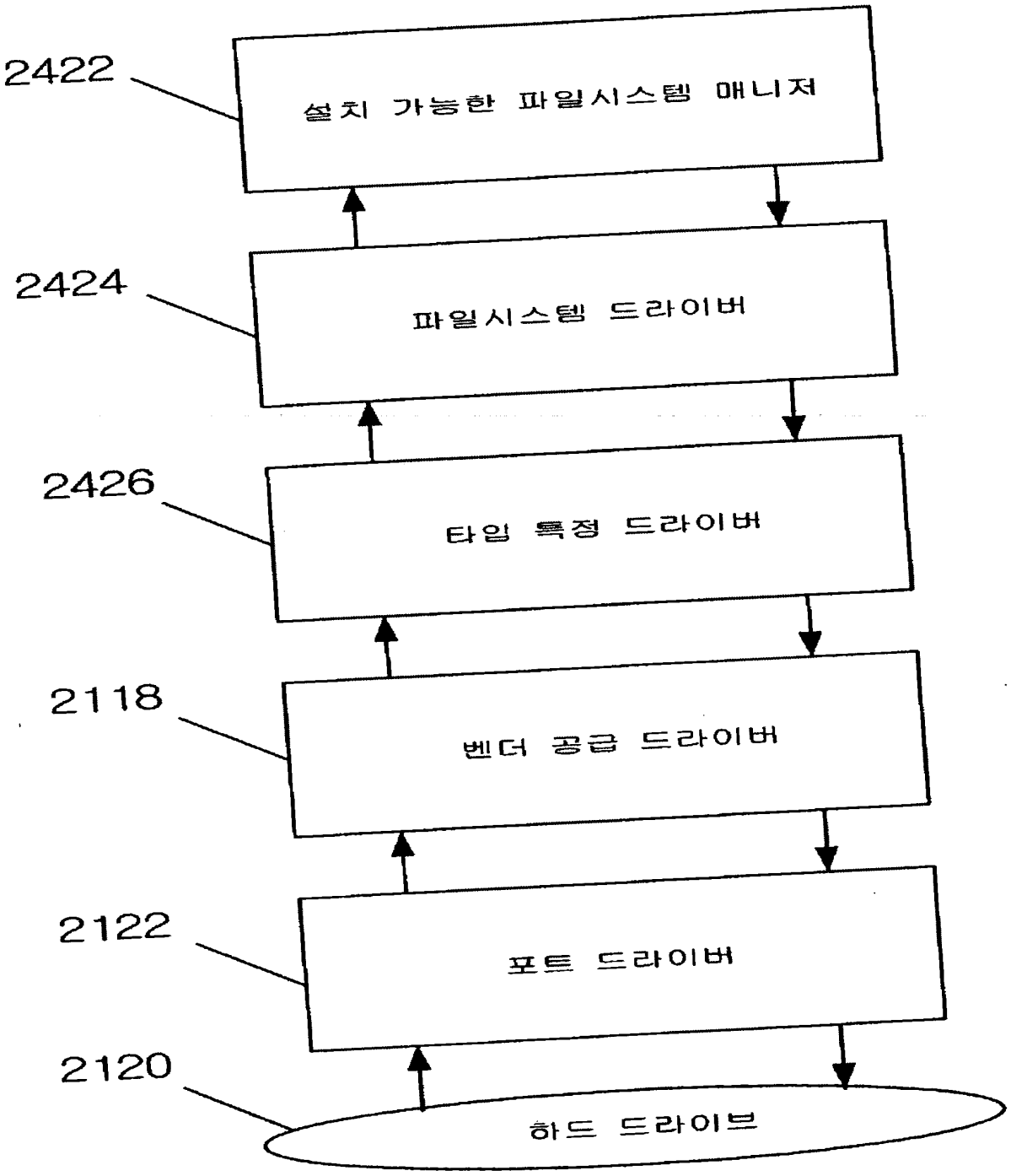
도면 23


2210
파일시스템
호출차단
2212
호출 타입을
처리합니까?
아니오
2214
체인에서 최초 호출을
다음 드라이버로 통과
예
2216
호출 프로세싱 실행
2218
시스템 애플리케이션
리턴

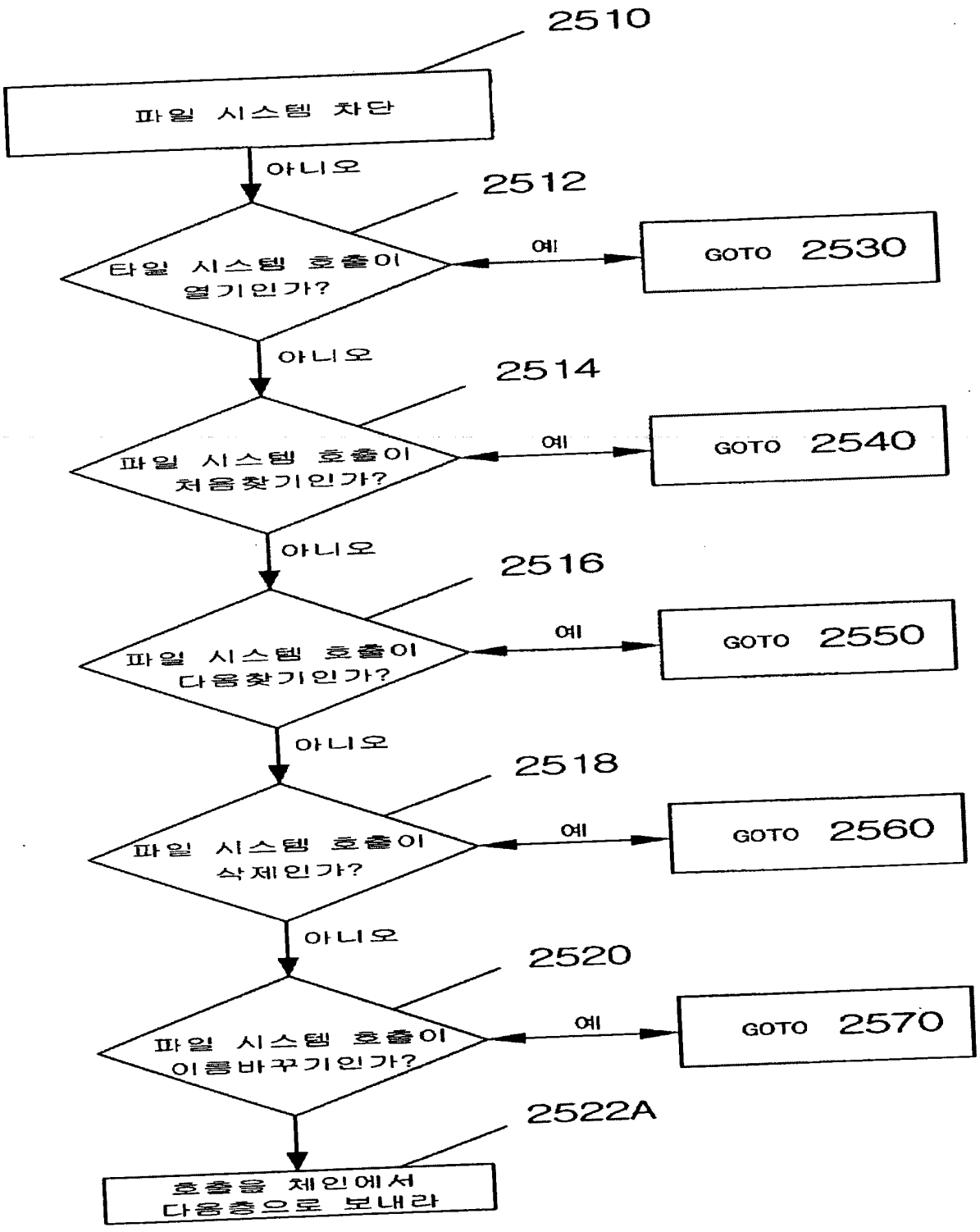
도면 24


2114
0
예비
2
설치 가능한 파일시스템 매니저
5
볼륨 트랙커
7
파일시스템 드라이버와
특정타입 드라이버
8
벤더 공급 드라이버
11
스카시-아이저
12
벤더 공급 드라이브
예비
19
여러가지 포트 드라이버
20
스카시 미니포트와 포트 드라이버
22
하드디스크 포트 드라이버
예비
25
플로피 포트 드라이버
예비
실제-모드 매퍼
29
예비

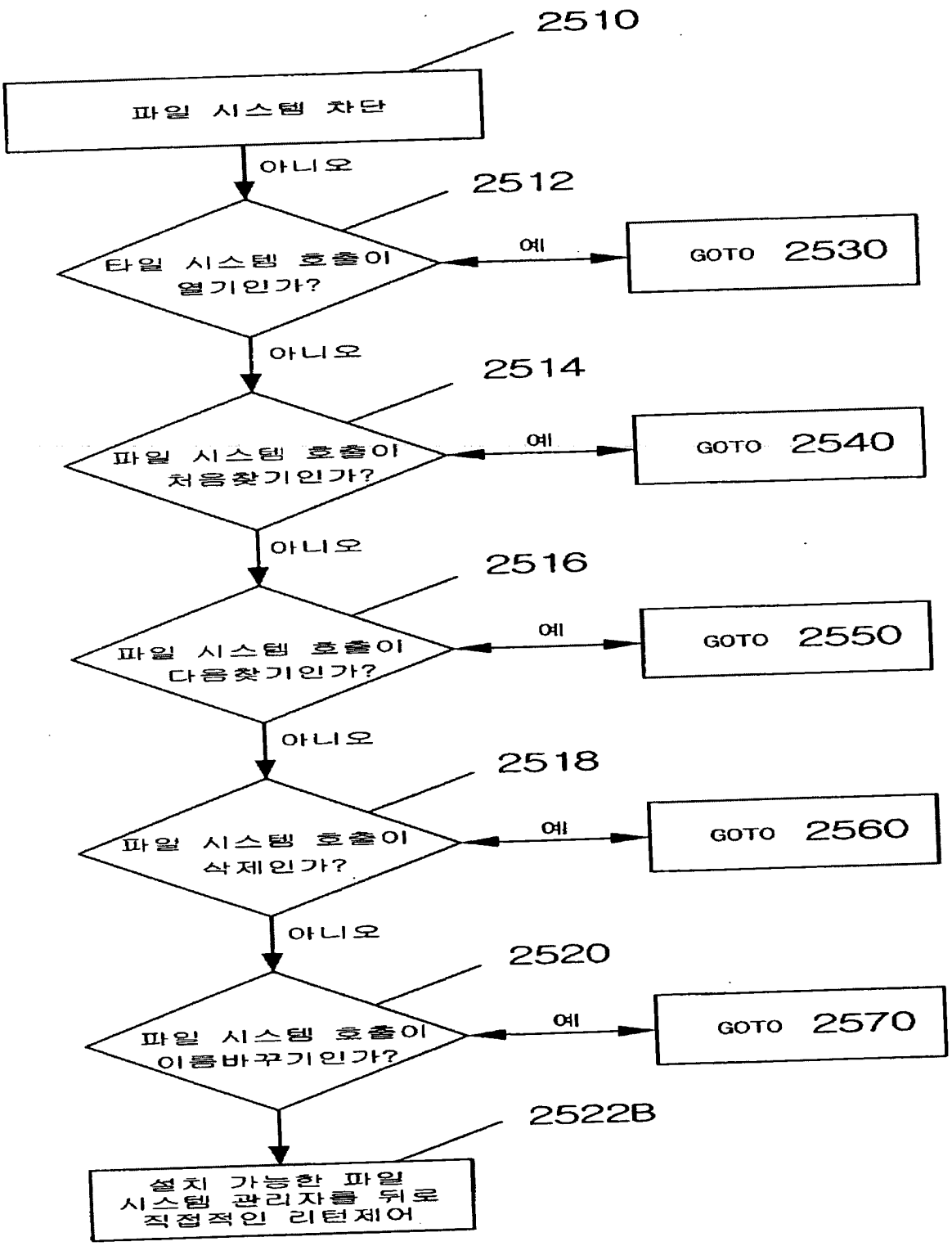
도면 25


2422
설치 가능한 파일시스템 매니저
2424
파일시스템 드라이버
2426
타입 특정 드라이버
2118
벤더 공급 드라이버
2122
포트 드라이버
2120
하드 드라이브

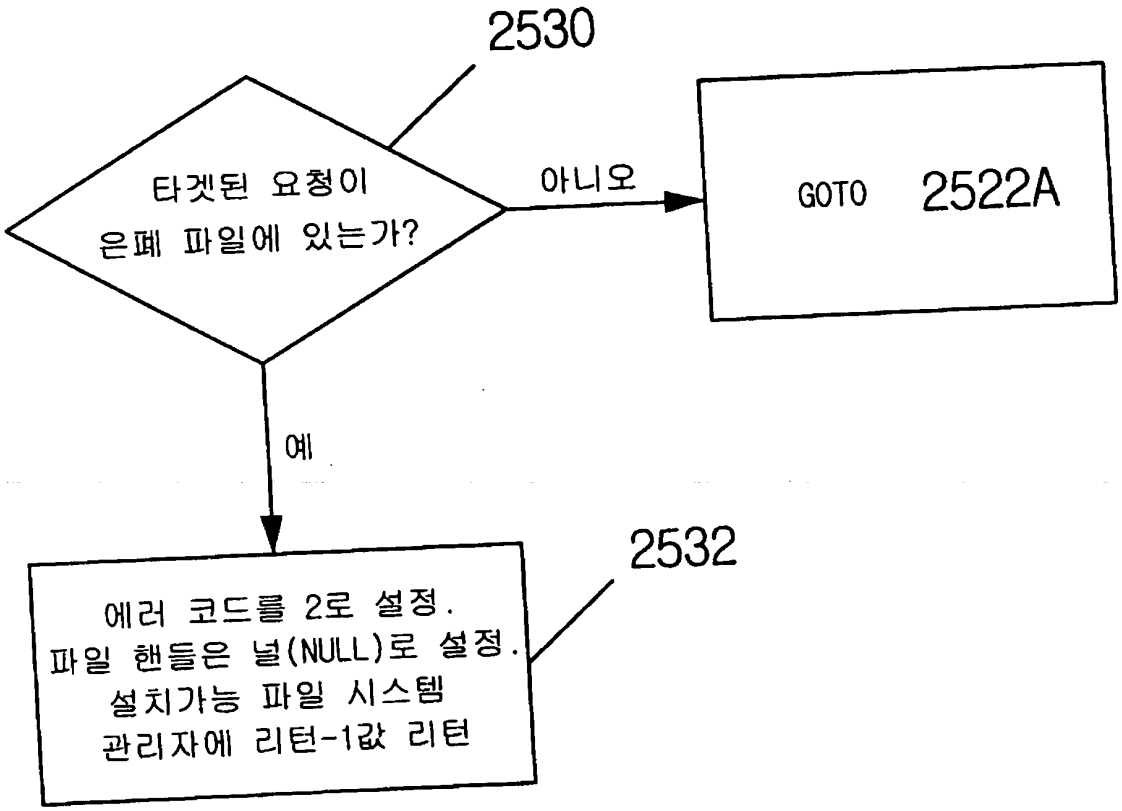
도면 26a


2510
파일 시스템 차단
아니오
2512
타일 시스템 호출이 열기인가?
예
GOTO 2530
아니오
2514
파일 시스템 호출이 처음찾기인가?
예
GOTO 2540
아니오
2516
파일 시스템 호출이 다음찾기인가?
예
GOTO 2550
아니오
2518
파일 시스템 호출이 삭제인가?
예
GOTO 2560
아니오
2520
파일 시스템 호출이 이름바꾸기인가?
예
GOTO 2570
2522A
호출을 체인에서 다음층으로 보내라

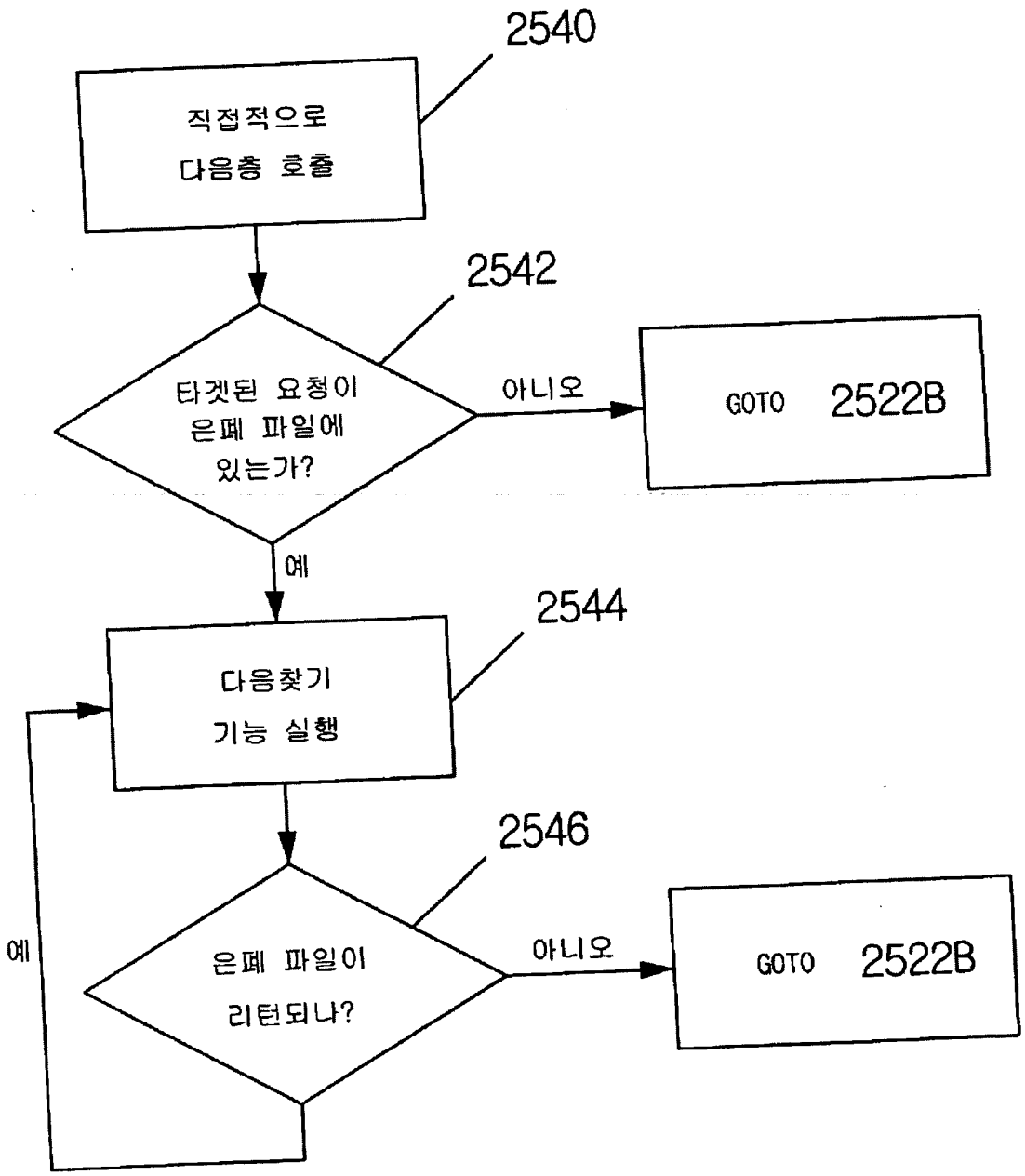
도면 26b


2510
파일 시스템 차단
아니오
2512
타일 시스템 호출이 열기인가?
예
GOTO 2530
아니오
2514
파일 시스템 호출이 처음찾기인가?
예
GOTO 2540
아니오
2516
파일 시스템 호출이 다음찾기인가?
예
GOTO 2550
아니오
2518
파일 시스템 호출이 삭제인가?
예
GOTO 2560
아니오
2520
파일 시스템 호출이 이름바꾸기인가?
예
GOTO 2570
2522B
설치 가능한 파일 시스템 관리자를 뒤로 직접적인 리턴제어

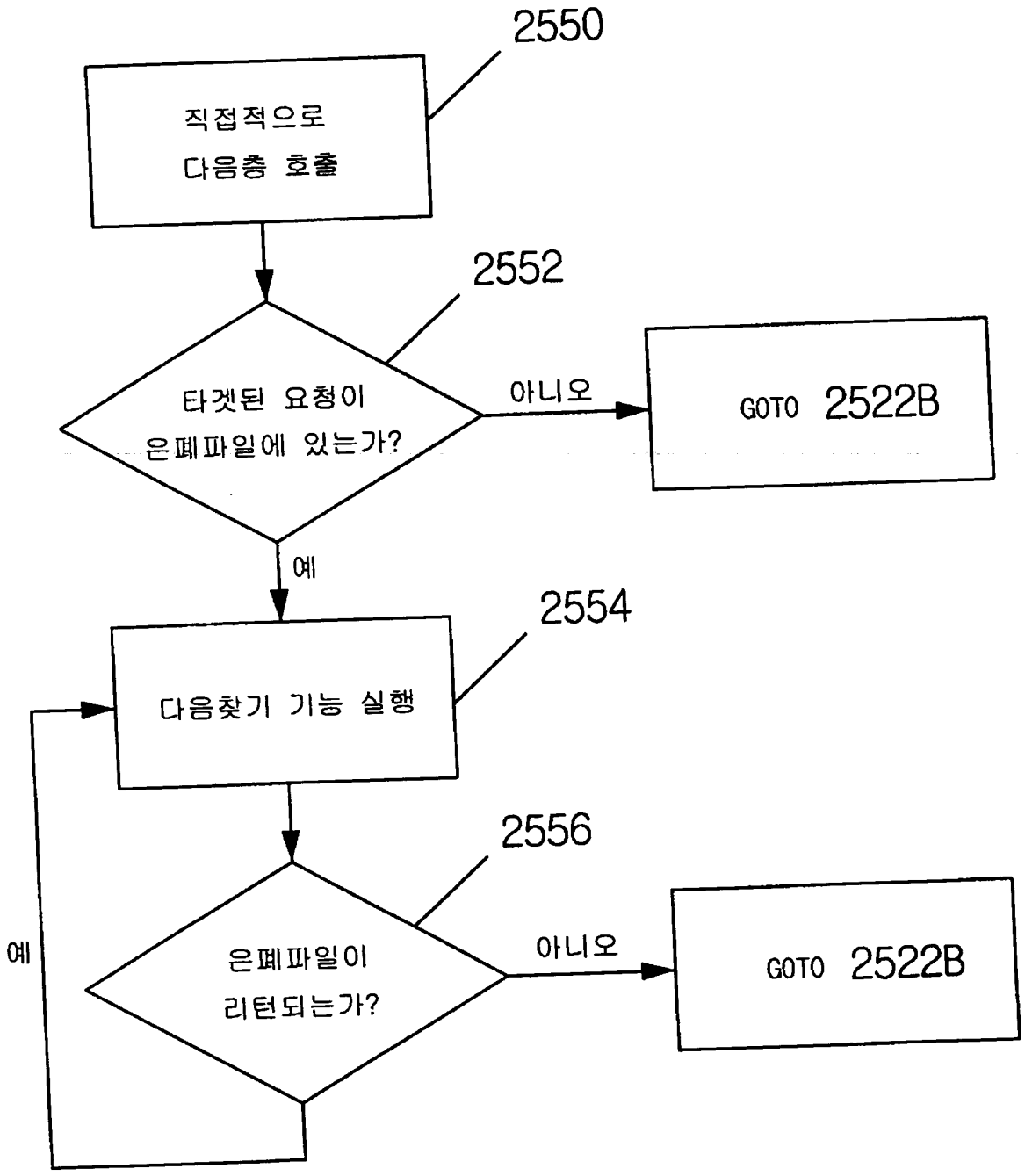
도면 26c


2530
타겟된 요청이
은폐 파일에 있는가?
아니오
GOTO 2522A
예
2532
에러 코드를 2로 설정.
파일 핸들은 널(NULL)로 설정.
설치가능 파일 시스템
관리자에 리턴-1값 리턴

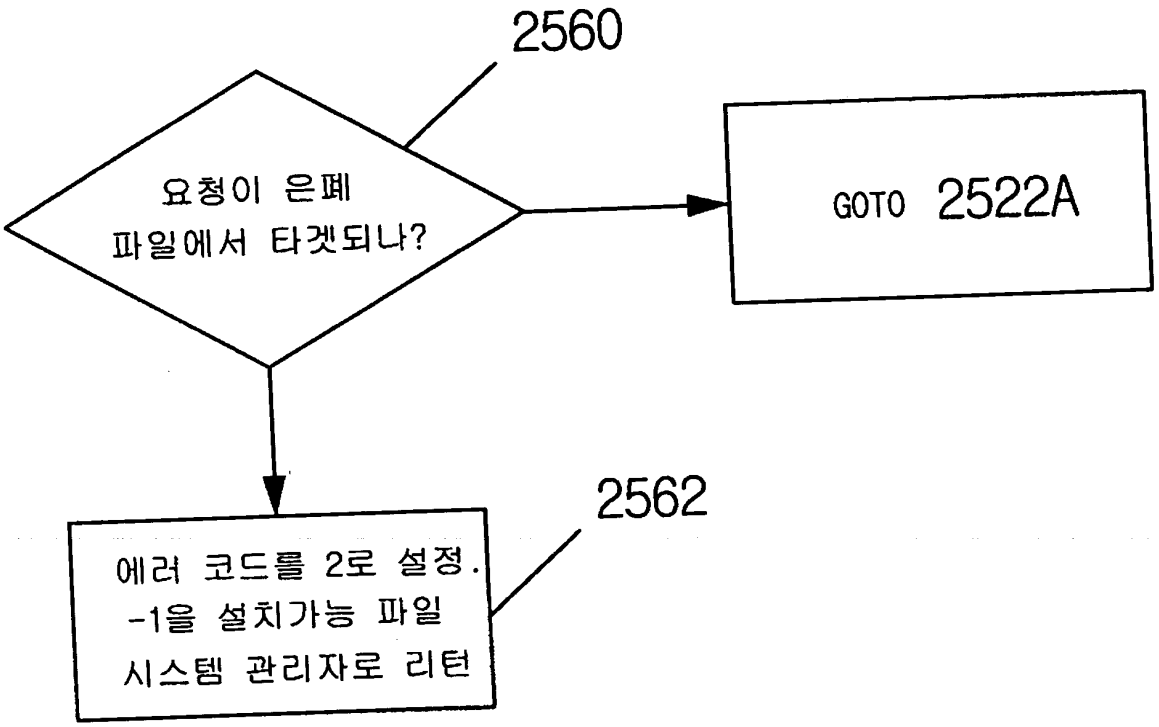
도면 26d


2540
직접적으로
다음층 호출
2542
타겟된 요청이
은폐 파일에
있는가?
아니오
GOTO 2522B
예
2544
다음찾기
기능 실행
2546
은폐 파일이
리턴되나?
아니오
GOTO 2522B
예

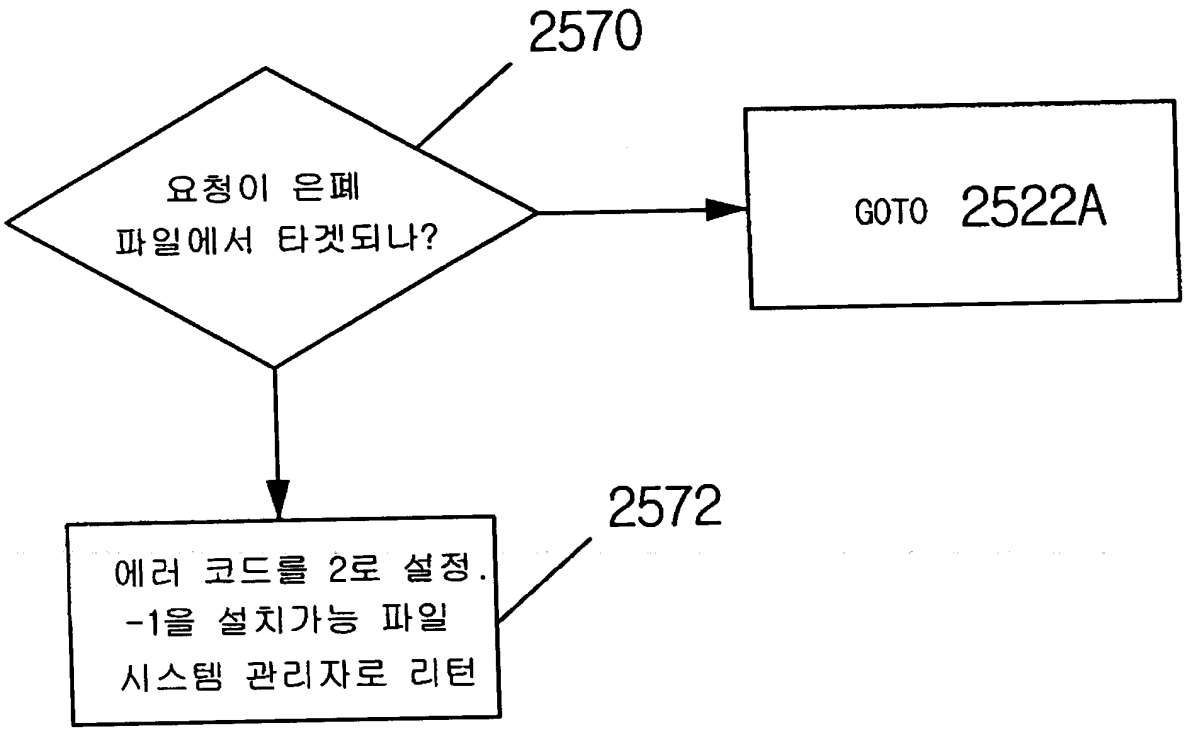
도면 26e


2550
직접적으로
다음층 호출
2552
타겟된 요청이
은폐파일에 있는가?
아니오
GOTO 2522B
예
2554
다음찾기 기능 실행
2556
은폐파일이
리턴되는가?
아니오
GOTO 2522B
예

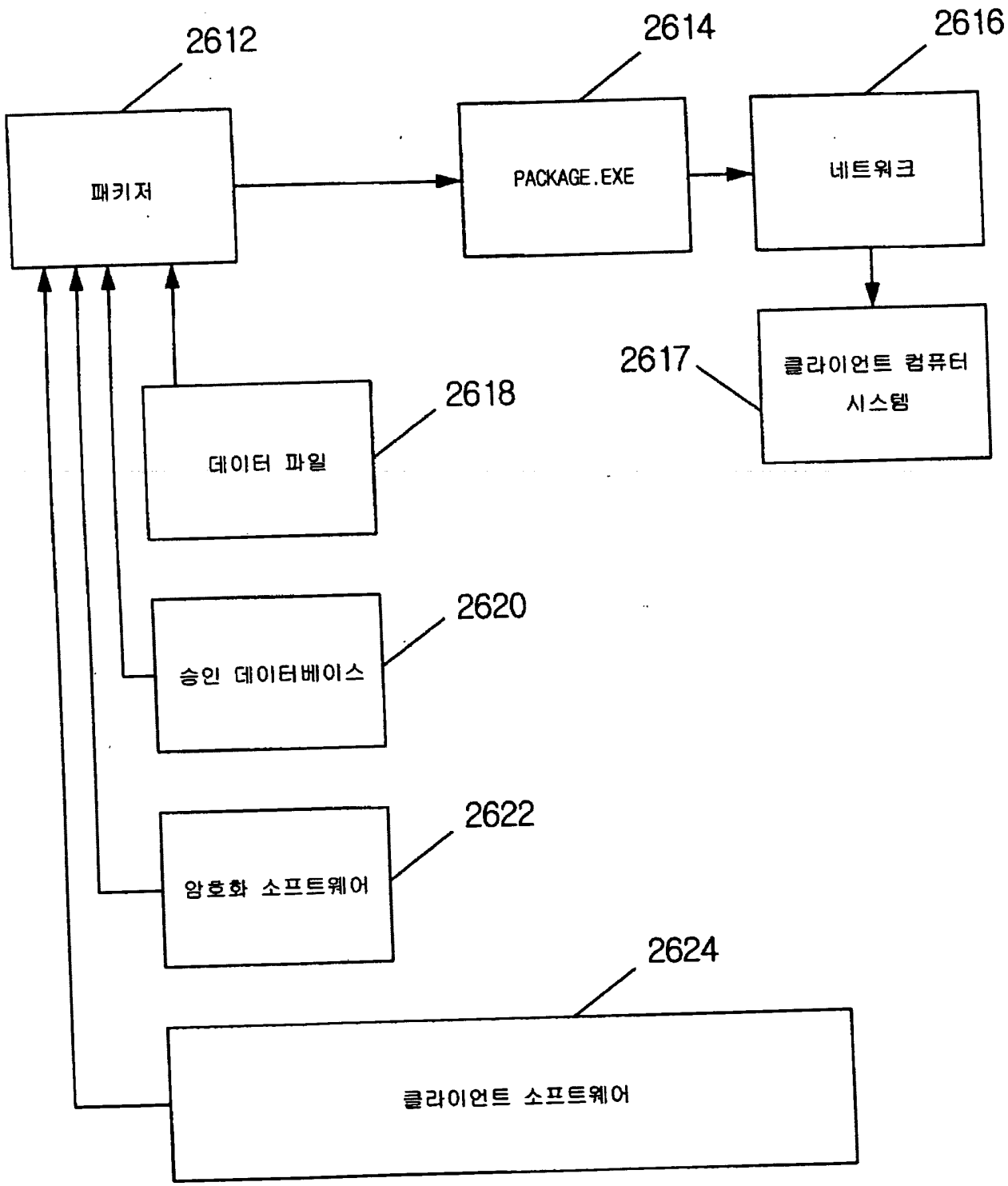
도면 26f


2560
요청이 은폐
파일에서 타겟되나?
GOTO 2522A
2562
에러 코드를 2로 설정.
-1을 설치가능 파일
시스템 관리자로 리턴

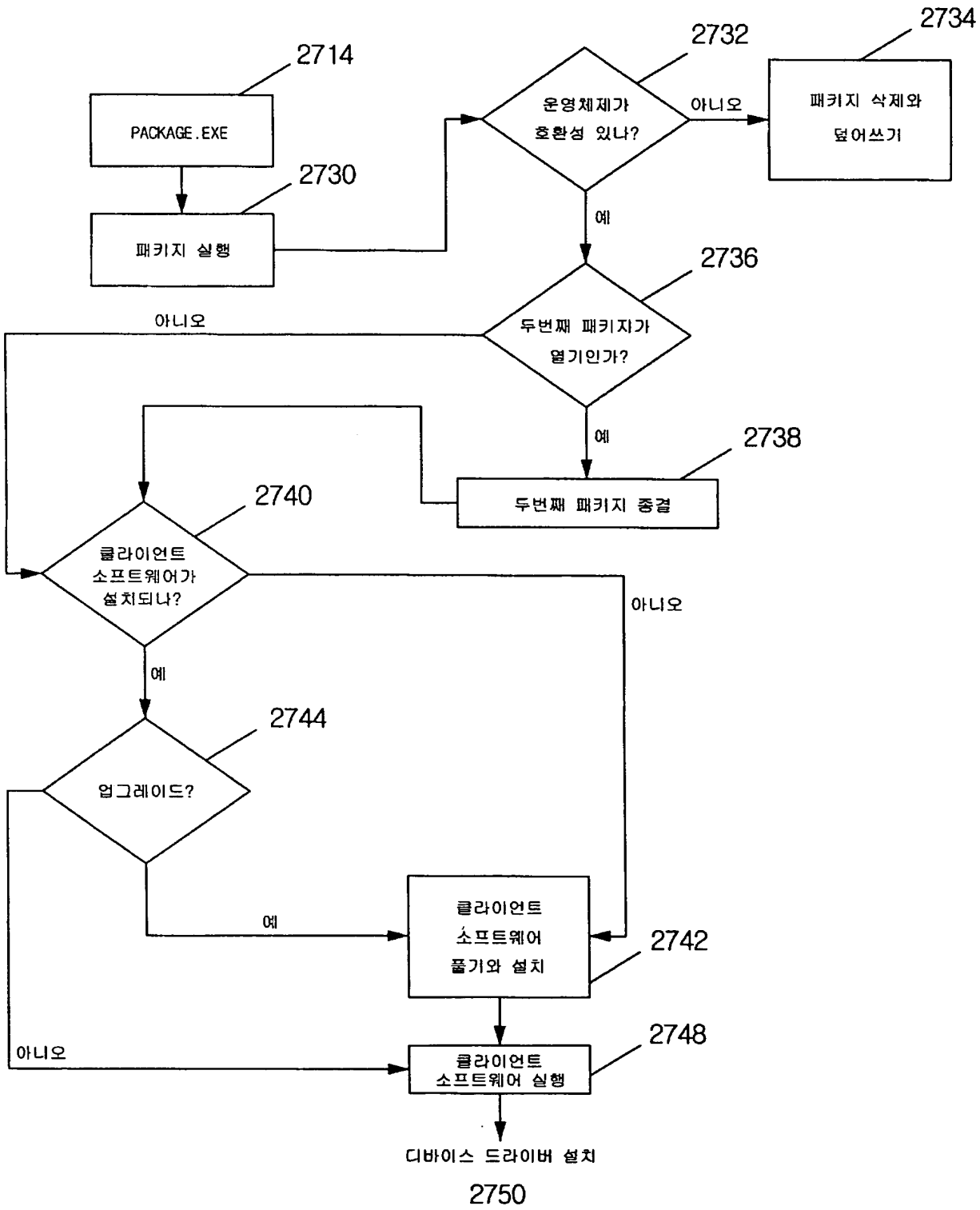
도면 26g


2570
요청이 은폐
파일에서 타겟되나?
GOTO 2522A
2572
에러 코드를 2로 설정.
-1을 설치가능 파일
시스템 관리자로 리턴

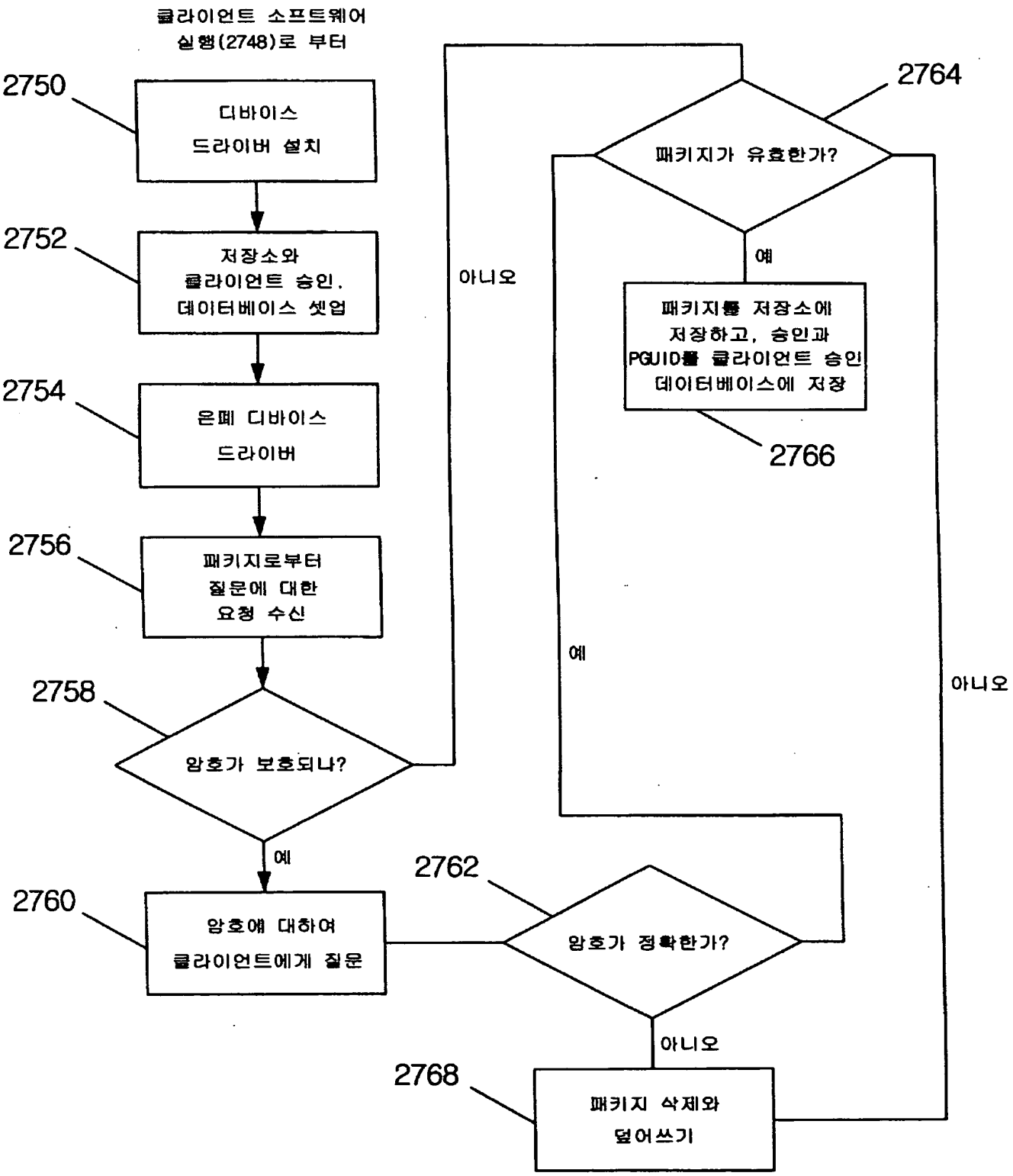
도면 27


2612
패키저
2614
PACKAGE.EXE
2616
네트워크
2617
클라이언트 컴퓨터 시스템
2618
데이터 파일
2620
승인 데이터베이스
2622
암호화 소프트웨어
2624
클라이언트 소프트웨어

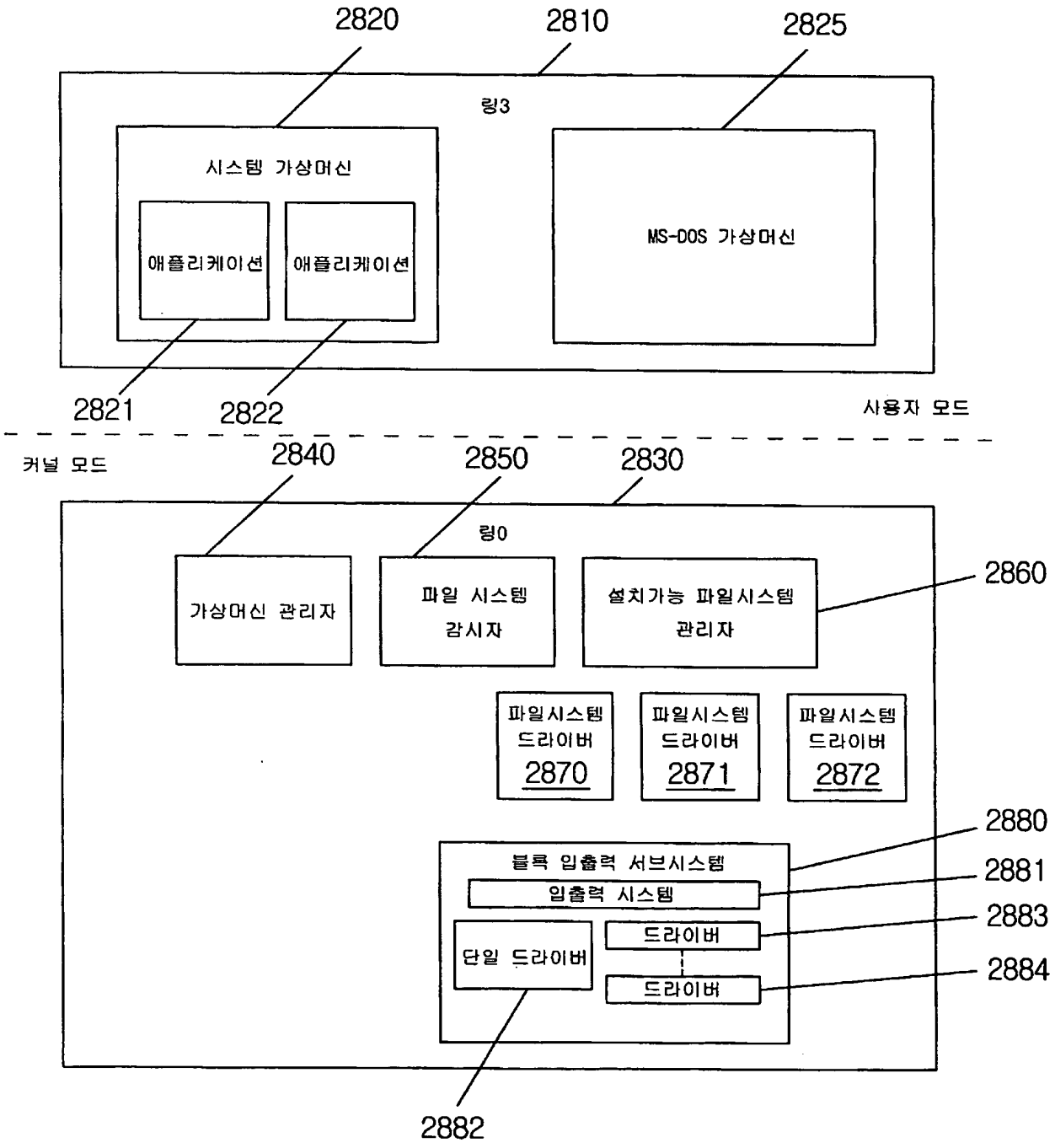
도면 28a


2714
PACKAGE.EXE
2730
패키지 실행
2732
운영체제가 호환성 있나?
아니오
2734
패키지 삭제와 덮어쓰기
예
2736
두번째 패키지가 열기인가?
아니오
예
2738
두번째 패키지 종결
2740
클라이언트 소프트웨어가 설치되나?
아니오
예
2744
업그레이드?
예
2742
클라이언트 소프트웨어 풀기와 설치
아니오
2748
클라이언트 소프트웨어 실행
디바이스 드라이버 설치
2750

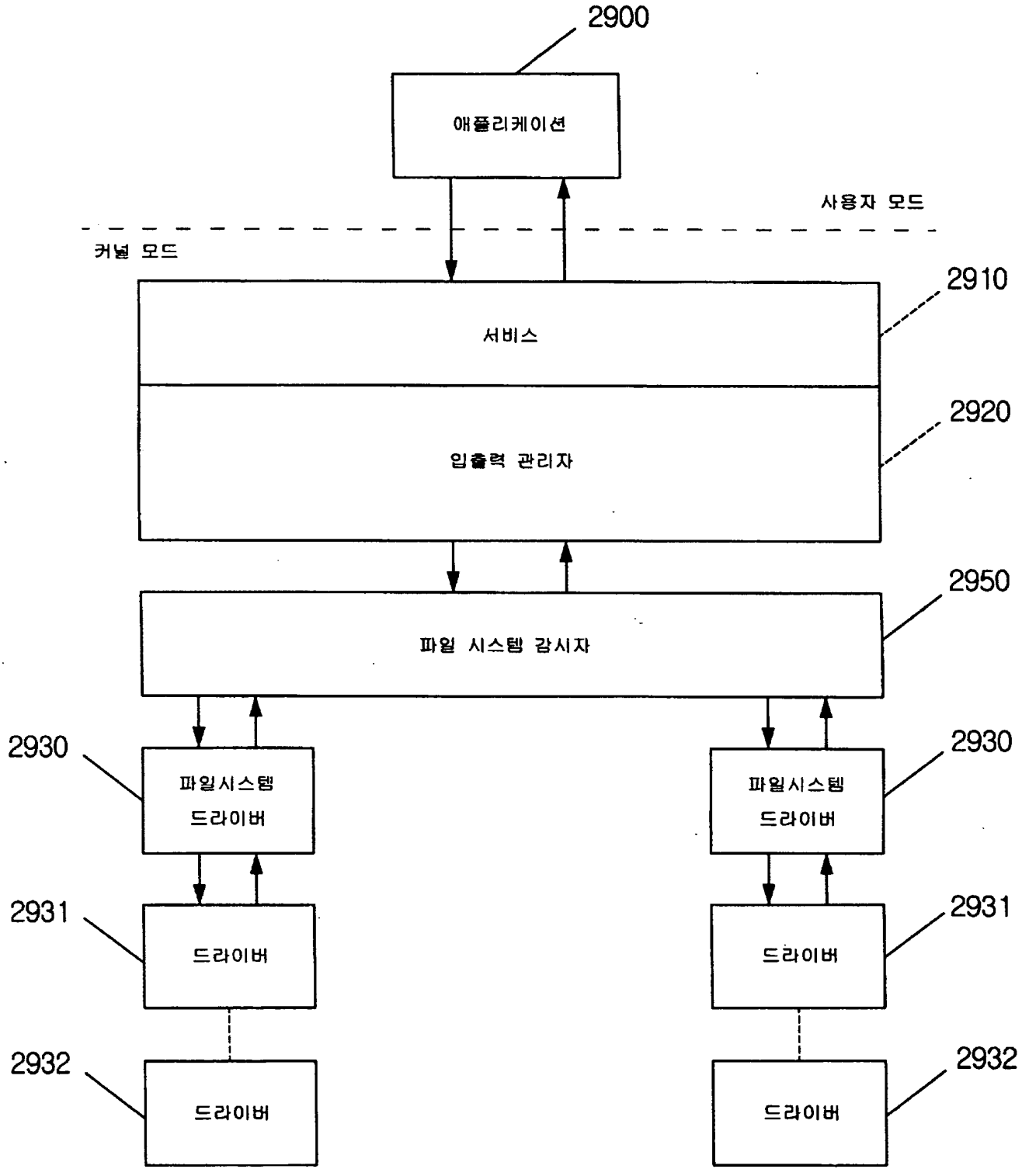
도면 28b


클라이언트 소프트웨어
실행(2748)로 부터
2750
디바이스
드라이버 설치
2752
저장소와
클라이언트 승인,
데이터베이스 셋업
2754
은폐 디바이스
드라이버
2756
패키지로부터
질문에 대한
요청 수신
2758
암호가 보호되나?
예
아니오
2760
암호에 대하여
클라이언트에게 질문
2762
암호가 정확한가?
아니오
2764
패키지가 유효한가?
예
아니오
2766
패키지를 저장소에
저장하고, 승인과
PGUID를 클라이언트 승인
데이터베이스에 저장
2768
패키지 삭제와
덮어쓰기

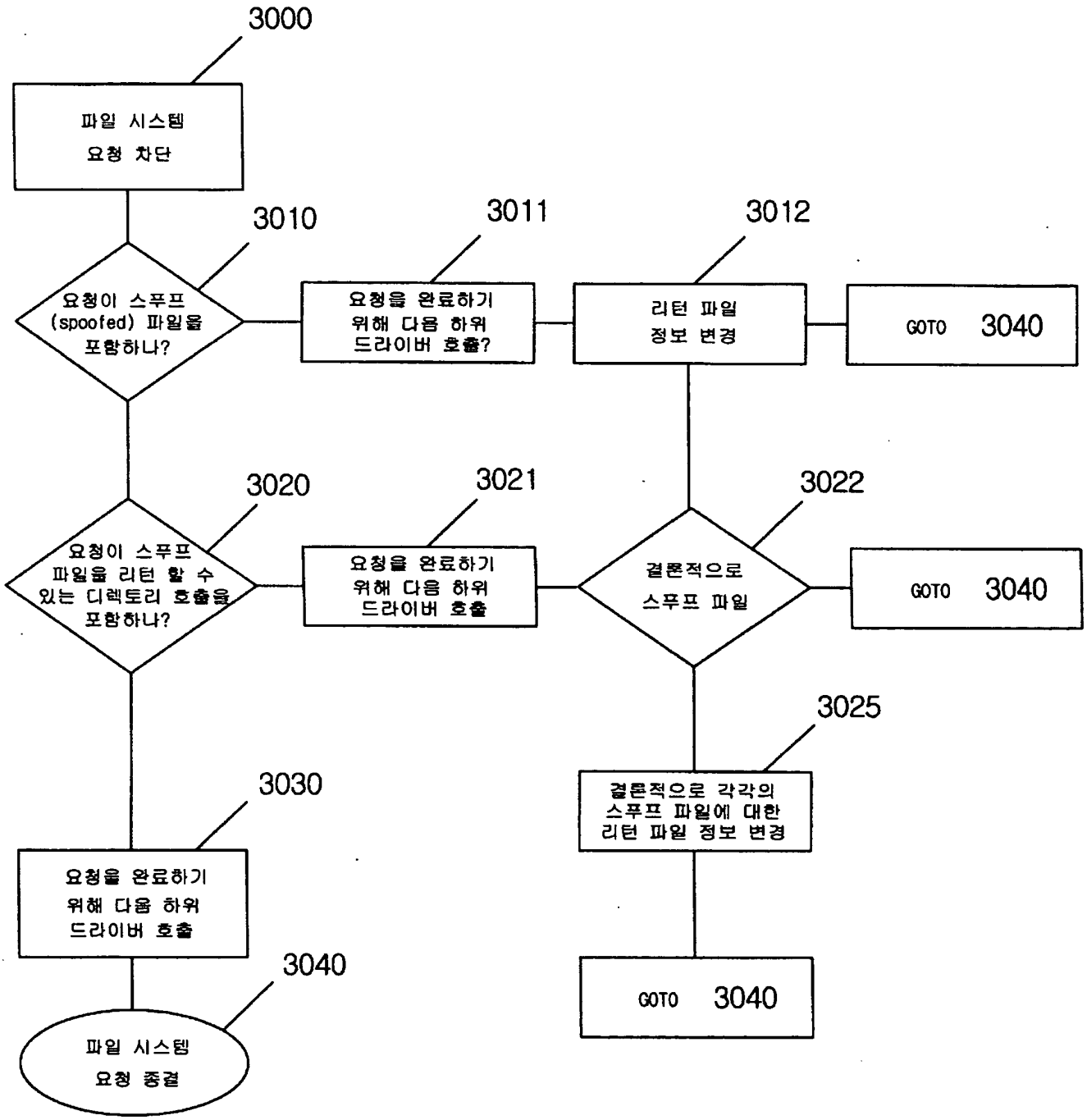
도면 29


2820
2810
2825
링3
시스템 가상머신
애플리케이션
애플리케이션
MS-DOS 가상머신
2821
2822
사용자 모드
커널 모드
2840
2850
2830
링0
가상머신 관리자
파일 시스템 감시자
설치가능 파일시스템 관리자
2860
파일시스템 드라이버 2870
파일시스템 드라이버 2871
파일시스템 드라이버 2872
2880
블록 입출력 서브시스템
입출력 시스템
2881
2883
드라이버
단일 드라이버
드라이버
2884
2882

도면 30


2900
애플리케이션
사용자 모드
커널 모드
2910
서비스
2920
입출력 관리자
2950
파일 시스템 감시자
2930
파일시스템
드라이버
2930
파일시스템
드라이버
2931
드라이버
2931
드라이버
2932
드라이버
2932
드라이버

도면 31


3000
파일 시스템
요청 차단
3010
요청이 스푸프
(spoofed) 파일을
포함하나?
3011
요청을 완료하기
위해 다음 하위
드라이버 호출?
3012
리턴 파일
정보 변경
GOTO 3040
3020
요청이 스푸프
파일을 리턴 할 수
있는 디렉토리 호출을
포함하나?
3021
요청을 완료하기
위해 다음 하위
드라이버 호출
3022
결론적으로
스푸프 파일
GOTO 3040
3025
결론적으로 각각의
스푸프 파일에 대한
리턴 파일 정보 변경
3030
요청을 완료하기
위해 다음 하위
드라이버 호출
3040
파일 시스템
요청 종결
GOTO 3040